

TDK-dolgozat

Czuprák Zsolt  
Éva Róbert  
Abonyi Gergő

# **Tudományos Diákköri Dolgozat**

Czuprák Zsolt                      Éva Róbert                      Abonyi Gergő  
(CZZPAAG.KEFO)    (EVRSAAG.KEFO) (ABGSAAG.KEFO)  
Mérnök informatikus hallgatók

## **FPGA képfeldolgozás alkalmazása hulladékgyűjtő roboton**

Témavezető:  
Dr. Kovács Tamás  
Főiskolai tanár

Kecskeméti Főiskola  
Gépipari és Automatizálási Műszaki Főiskolai Kar  
Kecskemét  
2012

## Tartalomjegyzék

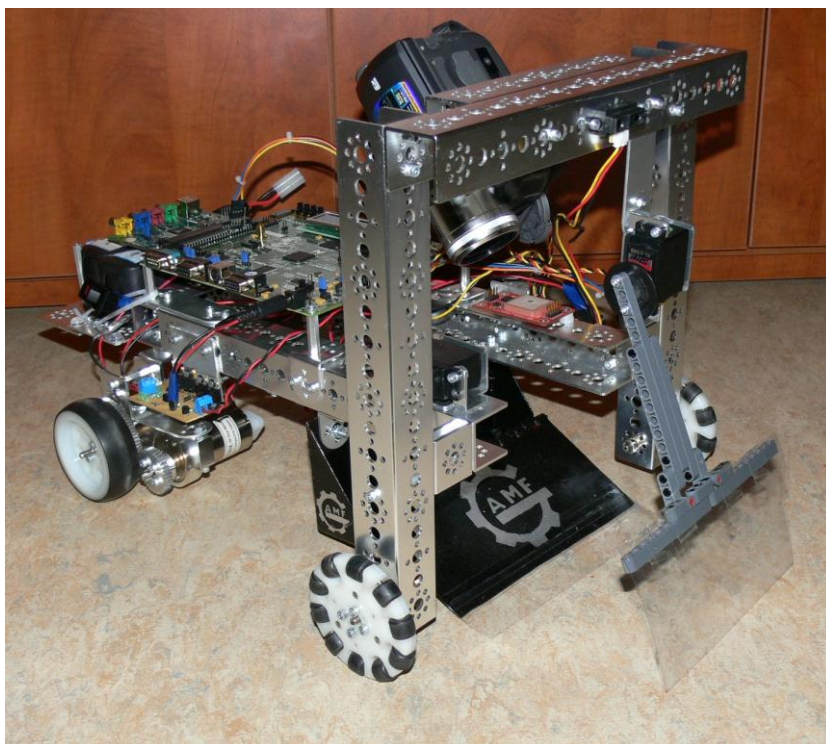
|                                                                    |    |
|--------------------------------------------------------------------|----|
| 1. Feladat ismertetése .....                                       | 5  |
| 2. Elméleti bevezető.....                                          | 6  |
| 2.1. A robottechnika világunkban .....                             | 6  |
| 2.2. Képfeldolgozási eljárások a kitűzött feladat szempontjából .. | 8  |
| Éldetektálás.....                                                  | 9  |
| A felbontás csökkentése .....                                      | 9  |
| Színkoordináta rendszerek.....                                     | 10 |
| A színek azonosítása.....                                          | 12 |
| Képjavítás „nyitás” és „zárás” eljárással.....                     | 12 |
| Egyszerű mintázat azonosítás (Pattern matching) .....              | 13 |
| A lényeges terület megkeresése .....                               | 14 |
| 2.3. A robot központi egysége – az FPGA.....                       | 15 |
| A Hirose FX2 interfész.....                                        | 18 |
| Analóg digitális konverter .....                                   | 18 |
| 3. A robot komponensei.....                                        | 20 |
| A robot vázának kialakítása.....                                   | 20 |
| Objektumérzékelés .....                                            | 21 |
| VDEC 1 típusú video dekóder modul .....                            | 21 |
| A robot irányítása .....                                           | 22 |
| Vezérlőpanel .....                                                 | 23 |
| Elembegyűjtő mechanizmus.....                                      | 23 |
| Szervomotorok működtetése .....                                    | 24 |
| Robot menetszabályzása.....                                        | 24 |
| 4. Az FPGA-ra fejlesztett alkalmazások .....                       | 26 |
| 4.1. Terület bejárás .....                                         | 26 |

|                                                                     |    |
|---------------------------------------------------------------------|----|
| 4.2. A képfeldolgozó egység .....                                   | 27 |
| Egyszerű azonosítás szín és méretek alapján .....                   | 27 |
| Felbontás csökkentés és erózió.....                                 | 30 |
| Kép deformáció és korrekciók .....                                  | 33 |
| Elforgatás transzformáció.....                                      | 35 |
| Alakzat-felismerés mintázat illesztéssel.....                       | 36 |
| Az alakzat-felismerési eljárás kidolgozása az FPGA-n .....          | 42 |
| 5. A projekt jelenlegi állapota és a továbbfejlesztési tervek ..... | 43 |

## 1. Feladat ismertetése

Ebben a dolgozatban bemutatásra kerül egy olyan autonóm FPGA által vezérelt robot kifejlesztése, amely képes egy adott területet bejárni és közben az előre kalibrált színű vagy formájú kisebb elemeket képfeldolgozás alapján felismerni, felszedni a földről és begyűjteni azt. A képfeldolgozás a mi esetünkben határozott kontúrral rendelkező homogén (egyszínű) tárgyra, valamint enyhén zajos élénk színektől mentes, tipikusan padló háttérre vonatkozik. A robot képes kikerülni az elé helyezett vagy útjába eső akadályokat, tárgyakat valamint megoldja a terület bejárás során jelentkező *sarokhelyzeteket* is. A robot olyan ipari környezetben hasznosítható, ahol egy gyártási folyamat során keletkező különböző színű, formájú lehulló hulladékot szeretnénk megadott feltételek alapján begyűjteni, valamilyen felhasználási vagy újrahasznosítási célból. A felhasználásra a gyárban lehulló összes elem alkalmatlan, ezért szelektálni kell azt. A legköltséghatékonyabb megoldás, ha már a begyűjtéskor megtörténik a szelektálás és így csak a megfelelő elem kerül begyűjtésre. A lehullott hulladékok a gyár teljes területén szétszórtnan helyezkedhetnek el.

A bemutatásra kerülő robot első verzióját a *Digilent Inc.* által minden évben megszervezésre kerülő *Digilent Design Contest 2012* –es nemzetközi elektronikai terméktervező versenyére készítettük el.



1.-1. ábra: A képen az elkészült robot látható.

## 2. Elméleti bevezető

### 2.1. A robottechnika világunkban

A robottechnika, mint fogalom az 1970-es években jelent meg és hamar önálló tudományterületté nőtte ki magát. A műszaki tudománynak ezen ága több szálon kapcsolódik más tudományágakhoz, mint például informatika és a matematika. A robotika megjelenik az iparban és a kutatásokban egyaránt, illetve fejlesztések folynak a mentőterepkutató és egyéb irányokba is. A robottechnika a termelő iparban, mint eszköz a termelési folyamatok automatizálására fejlődött ki. Az ipari termelés nagyságának és gyorsaságának növekedését eredményezi, ami a fejlett ipari államoknak köszönhetően a robotika gyors fejlődését elősegíti. A robotika a veszélyes és egészségre ártalmas helyeken az emberi munka kiváltására is alkalmas.

A mai állapot a robottechnika három generációja alatt fejlődött ki. Jelenleg bizonyos robotok mesterséges intelligenciával (intelligens robotok) rendelkeznek, így emberi beavatkozás nélkül is képesek teljes mértékben alkalmazkodni a feladat körülményeihez és elvégezni azt. Az ilyen mesterséges intelligenciával ellátott robotok vezérléséhez komoly erőforrással rendelkező modern számítógépek szükségesek.

A robottechnika ipari alkalmazása (2.1-1. ábra: *Ipari használatú robotkar.* utat nyitott az automata ipari rendszerek megvalósításához, aminek fontos gazdasági haszna is van. Az eszközt elég egyszer megvásárolni és nem vár rendszeres fizetést, a munkaköltségei is folyamatosan csökkennek. A robot alkalmazható az év minden napján a nap minden percében az eredeti magas minőség megtartásával.



2.1-1. ábra: Ipari használatú robotkar.

Az ipari alkalmazáson kívül napjainkban megjelentek az otthoni felhasználású háztartási és szórakoztató, valamint a speciális használatra készült robotok. A technológia felhasználása kiterjed a porszívózástól és a fűnyírástól egészen a robbantásokig vagy akár víz alatti kutatásokig (2.1-2. ábra: *Víz alatti kutató robot.*), de akadnak teszt jelleggel börtön rabokra vigyázó robotok is. A robottechnika rendkívül széles és gyorsan fejlődő szférája szinte minden háztartásba elér.



2.1-2. ábra: Víz alatti kutató robot.

A fűnyíró illetve a porszívó robotok az otthoni felhasználású robotok körében a legkelendőbbek, de ez az adat valószínűleg változni fog, ha elkezdenek terjedni az otthon védelmére szolgáló, és a fogyatékkal élőket segítő eszközök.

A robottechnika másik gyorsan fejlődő területe az oktatási célra használt robotok. Ilyen területen a legnagyobb népszerűségnek örvendenek a *Mindstorms* által fejlesztett *Lego NXT* és a *Pioneer 3-DX*. A hadiipar is előszeretettel használja a robottechnika által nyújtotta lehetőségeket, például távolról irányítható bomba megsemmisítő eszköz vagy pilóta nélküli lopakodó repülőgép formájában.

A robotokon a környezete érzékelésének céljából egyre több szenzort helyeznek el. Ezek a robotokra szerelt szenzorok lehetnek például távolságérzékelők, nyomásérzékelők, gázérzékelők, fényérzékelők, gyorsulásmérő, giroszkóp, nyomógombok és még számos érzékelő fajta. Manapság az ipari robottechnikában számos esetben szerelnek a robotokra humanoid érzékszervekhez hasonló érzékelőket. Ilyen esetekben a robotot akár a látással vagy hallással is felruházzák. Az, hogy a robot mit csináljon a látottak illetve hallottak alapján, azaz milyen intelligenciája legyen a robotnak, azt már a rátöltött szoftverrel adhatjuk meg. Ilyen látással rendelkező robot például, az ipari gyártósoron dolgozó robot. Ez a robot figyeli a futószalagon érkező termékeket és a hibás termékeket kiválogatja a többi közül.

## 2.2. Képfeldolgozási eljárások a kitűzött feladat szempontjából

A képfeldolgozás minden formája egy végső szinten digitális jelfeldolgozás, amelyben a bemenet egy kép, mint például egy fénykép és a kimenete lehet akár szintén kép, vagy egy a képre vonatkozó paraméter lista. (A képfeldolgozás kifejezés általában a digitális képfeldolgozásra utal, de az optikai és az analóg képfeldolgozást is jelentheti. A mi esetünkben a képfeldolgozás a digitális feldolgozásra utal.)

A digitális feldolgozási technikák lehetnek kép javítási (*image enhancement*), a kép rekonstrukciós (*image restoration*), képtömörítési (*image compression*) és alakzat felismerési (*pattern recognition*) eljárások.

A kép javítása csoportba tartoznak a kép hangsúlyozásával, élesítésével foglalkozó eljárások. Ezek az eljárások a kijelzőn megjelenő kép hasznosabbá, szebbé tételében segítenek. Az eljárás során készült új kép nem tartalmaz se több, se kevesebb információt, mint amit maga az eredeti kép tartalmazott.

A kép rekonstrukciója eltér a képjavító módszerektől, célja a kép készülése közben fellépő káros hatások csökkentése vagy megszüntetése. Ilyen hatás lehet például a fényképezőgép érzékelője által felfogott zaj, fényképezőgép mozgása, de akár a légköri turbulencia is.

A képtömörítési eljárások célja, hogy csökkentsék az úgynevezett „lényegtelen” adatokat és megszüntesse a redundanciát, valamint a kép hatékonyabb tárolási módja. A tömörítési eljárás lehet veszteséges és veszteségmentes. A veszteségmentes tömörítés előnyös archiválási célból, valamint gyakran alkalmazzák orvosi képeknél, műszaki rajzoknál. A veszteséges módszerek alkalmasak a természeti képeknél, ahol kisebb veszteség még elfogadható és néha észrevehetetlen is.

Az alakzat-felismerés az a tudományterület, ami objektumok valamely kategóriába vagy osztályba sorolásával foglalkozik. A fogalmat, ha szélesebb értelemben vesszük, akkor a térbeli alakzatok, síkbeli képek, rajzok, írás, élőbeszéd, ujjlenyomat felismerést is magában foglalja.

A kitűzött feladathoz több képjavítási és képfeldolgozási módszert használtunk fel. Ezek a módszerek a végső pontos felismeréshez elengedhetetlenek. A képfeldolgozás hatalmas területet ölel fel, ezért a következőkben csak az általunk felhasznált eljárások, algoritmusok kerülnek bemutatásra.

## *Éldetektálás*

Élnek nevezzük azon pixelek halmazát, amely körül a kép tulajdonságai szignifikánsan megváltoznak. Ez általában a képen szereplő alakzatok széleinél, határvonalainál fordul elő, ez lehetőséget ad az alakzatok felismerésére is. A képfeldolgozás világában ezt a módszert kontúralapú szegmentálásnak nevezzük, melynek fontos lépése az élek detektálása. Az élek mentén az intenzitásfüggvény gyorsan változik, ezért a deriválás segítségével megkereshetjük az éleket. Az intenzitásfüggvény első és második deriváltja is fontos adatokat hordoz az élekről. Az éldetektálásról valamit a képek élesítéséről [7] –es referenciában részletesebben olvashatunk.

Az *OpenCV* könyvtár a *Canny* által kidolgozott éldetektálási módszert tartalmazza, az eljárás három fő kritériumra épül. A képen található élek ne vesszenek el, az él valós és detektált helye között csak minimális különbség legyen, valamint egy él csak egyszer legyen detektálva.

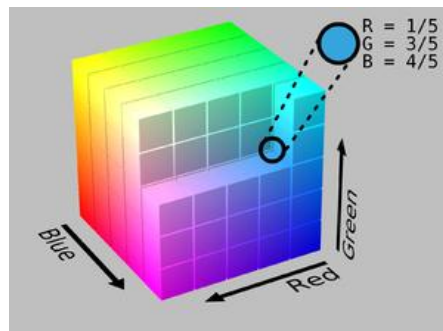
Első lépésben egy két-dimenziós Gauss szűrő segítségével, kiszűrjük a képen található zajokat, majd második lépésként a zajmentesített képen végigtolunk egy differenciáló mátrixot. Így megkapjuk azokat a pontokat ahol a pixel érték ugrásszerűen változik. Harmadik lépésként eltávolítjuk azon pixeleket, amelyek nem részei egyetlen élnek sem. Az eljárás utolsó lépéseként ellenőrizzük, hogy a megmaradt élek valós élek-e, erre *Canny* két küszöbértéket javasolt, egy alsó és egy felső értéket. Ha a pixel intenzitása nagyobb, mint a felső küszöbérték, akkor elfogadjuk él-pixelként. Ha az alsó küszöbértéknél kisebb, akkor az adott pixelt töröljük. Amennyiben a két érték között van, csak abban az esetben tartjuk meg, ha a szomszédos pixelek, a felső küszöbértéknél nagyobb intenzitásúak.

## *A felbontás csökkentése*

Gyakran előfordul, hogy a képkészítő eszköz által előállított kép, feldolgozásra túl nagy felbontású, hiszen a kép méretével (azaz felbontásával) egyenes arányosan nő a feldolgozandó pixelek száma is. Ez egyszerűen belátható, hiszen a kép pixelszáma és a feldolgozási idő egyenes arányosságban állnak egymással. Ezért gyakran felbontás csökkentő eljárásnak vetik alá a képet. Az új kép pixeleit a régi kép környező pixelei alapján határozhatjuk meg. A leggyakrabban alkalmazott megoldás, hogy a képet pixel-blokkokra bontják fel és egy pixel-blokkhoz rendelik hozzá a módosított kép egyetlen pixelét. Ez történhet például a pixel értékek átlagolásával vagy egy komplikáltabb függvénnyel.

## Színkoordináta rendszerek

Az *RGB* három komponensből áll (*Red-Vörös*, *Green-Zöld*, *Blue-Kék*), minden komponens 8 bites (0-255). Az *RGB* színrendszert főképp a számítógépes színhasználatnál alkalmazzák. A három alapszín összeadásával állítja elő a színeket. Az *RGB* színrendszer egy adott koordinátája tartalmazza a színek intenzitását, azaz a színek intenzitása és maga a szín nem választhatóak szét. A környezeti fényhatások színtorzuláshoz vezethetnek, azaz egy előre bekalibrált szín, más fényhatások mellett az eredeti szín koordinátáitól jelentősen eltérhet.



2.2-1. ábra Az RGB színkoordináta rendszer

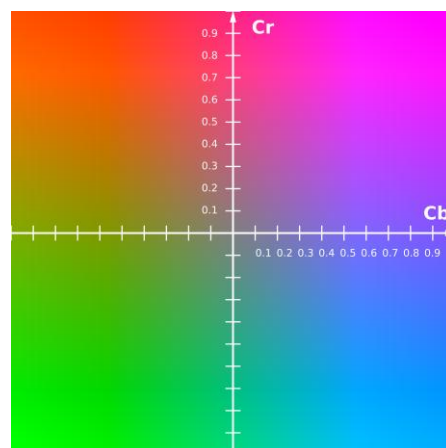
Az *YCrCb/YUV* színrendszerből *RGB* rendszerbe történő konvertálás a következő képletekkel adható meg:

$$R = 1.164(Y - 16) + 1.596(V - 128) \quad (2.2.1)$$

$$G = 1.164(Y - 16) - 0.813(V - 128) - 0.391(U - 128) \quad (2.2.2)$$

$$B = 1.164(Y - 16) + 2.018(U - 128) \quad (2.2.3)$$

Az *YCrCb* szín-mintavételezési eljárás. A 4:2:2 arány az *Y*, *Cr* és *Cb* komponensek egymáshoz viszonyított arányát mutatja, a fényességértékek (*Y*) tipikusan 8 biten (1 byte), a színkülönbségi jelek (*Cr*, *Cb*) 4-4 biten tárolhatók, így az egy képponthoz (pixel) tartozó információtartalom összesen 2 byte. Az *YCrCb* színkoordináta rendszere az 2.2-2. ábralátható.



2.2-2. ábra *YCrCb* színkoordináta rendszere

A *LUV* (*L* a Luminance vagy Luma,  $L = Y$ ) és *YIQ* színrendszerek az európai (*PAL*) és az amerikai (*NTSC*) televíziós szabványok és a digitális videók számára készültek. A *LUV/YIQ* szabványok sajátosságait az határozza meg, hogy a színes kép a fekete-fehér képernyőn is látható legyen. Ez a feltétel fontos volt a színes televíziók bevezetése során, ugyanakkor meg kellett oldani azt a problémát is, hogy a fekete-fehér televízió az azonos színű, de eltérő fényességű anyagok ugyanolyan szürkének látszanak.

Az *RGB* színrendszerből *YCrCb/YUV* színrendszerbe történő konvertálás a következő képletekkel adható meg:

$$Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16 \quad (2.2.4)$$

$$Cr = V = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128 \quad (2.2.5)$$

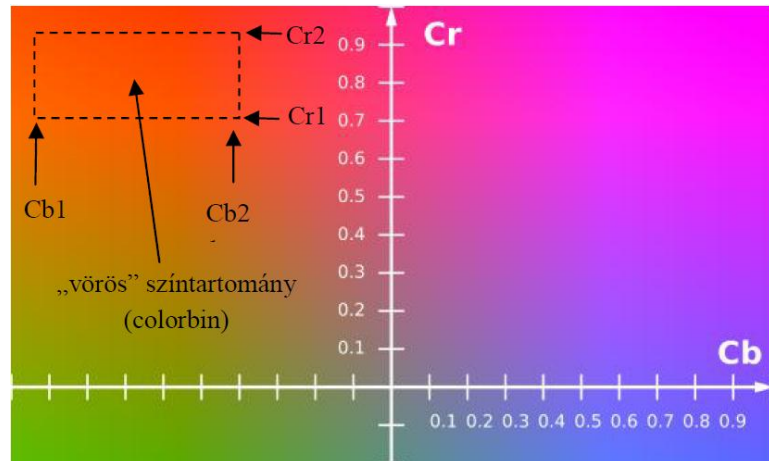
$$Cb = U = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128 \quad (2.2.6)$$

A fekete-fehér televíziókon csak az (*Y*) jel látható. A szín információkat két külön jel hordozza, a *PAL* illetve az *NTSC* szabványok különbségei miatt ezekhez eltérő sáv szélességek tartoznak.

A világosság adatok számára biztosított frekvencia a színek frekvenciájának kétszerese, mert a szürkeárnyalatok több információt tartalmaznak. Az *NTSC* rendszerben a vörös szín hangsúlyosabb (*I* tengely fontosabb), a rendszer a *PAL*-hoz képest kevesebb színt tartalmaz, de a nagyobb sáv szélesség miatt pontosabban vihetők át a színek. Így az emberi bőr szín valóság hűen jeleníthető meg, valamint a megjelenített kép élénkebb. A *PAL* rendszer színrendszere kiegyensúlyozottabb, az *U* és *V* tengelyek közel azonos szín tartományt fognak át, valamint a sáv szélességek is megegyeznek. A világosság jel jelentősége miatt a két televíziós szabvány szürkébb színeket eredményez, mint a számítógépekben alkalmazott *RGB* színrendszer. . A *YUV/YIQ* színtér foglalat a kék tengely irányában enyhén összenyomódott Descartes féle orthogonális koordináta rendszer, amelyet az *RGB* színek még így sem töltenek ki teljesen, hanem egy konvex poliéderen belül helyezkednek el. A szín tengelyekből (*I/Q* illetve az *U/V*) képzett jeleket, egymáshoz képest 90°-al elforgatva egy meghatározott frekvenciájú (*PAL* 4,43MHz, *NTSC* 3,58MHz) szinusz jelre ültetik, majd a két jelet egyesítik. Az egyesített jel neve *C* mint *Chroma*. Mindkét szabvány hátránya, hogy a jelek egyesítése miatt a színek torzulhatnak.

## A színek azonosítása

Az adott szín felismeréséhez a megadott szín koordinátáinak elmentése, majd a beérkező színek koordináták hozzá való hasonlításával oldható meg. Ennek a problémának a megoldása az úgynevezett tűréshatárral rendelkező színazonosítás. Ez a megoldás azt jelenti, hogy létrehozunk egy színtartományt (*colorbin*). Ez a színtartomány a konfigurált szín színárnyalatait tartalmazza, azaz a színtől nem sokban eltérő színek koordinátáit is még a bekonfigurált színeként ismeri fel a rendszer. Ezt a színek koordináta tartományt érdemes jól megválasztani, mert minél



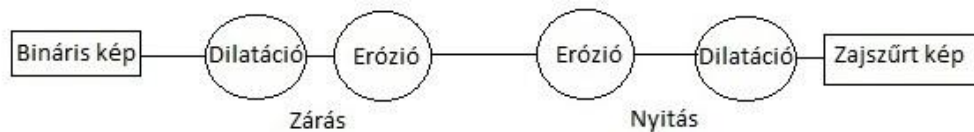
2.2-3. ábra Vörös szín felismerésére szolgáló színtartomány

nagyobbra választjuk meg a tartományt, annál több megfelelő pixel lesz. Azaz annál több zaj is előfordulhat a kapott képen. Ha viszont túl kicsire választjuk meg a tartományt, akkor viszont kevesebb hasznos pixelünk lesz, vagy akár elcsúszhatunk a lényeg felett. A 2.2-3. ábraán egy meglehetősen nagy színtartomány látható a vörös szín azonosítására. Ha a kérdéses pixel *Cb* koordinátája *Cb1* és *Cb2*, a *Cr* koordinátája pedig *Cr1* és *Cr2* közé esik, akkor mondjuk azt a pixelről, hogy adott vörös színű.

## Képjavítás „nyitás” és „zárás” eljárással

Amennyiben bináris képpel dolgozunk (azaz olyan képpel, amelyben csak fekete vagy fehér pixelek vannak), gyakran nagyon zavaró lehet a képen előforduló zaj. Ez a zaj kisebb nagyobb, de általában néhány pixel nagyságú foltokat jelent. Az ilyen foltok az alakzat felismerésben gondot okoznak, az alakzat felismerhetetlenné válhat, vagy akár más alakzatra is átfordulhat az eredeti tartalom. Emiatt további szűrésre, simításra van szükség. (Lásd például [5] –ös referencia.) A probléma megoldására egy kétlépcsős eljárást használnak. Elsőként egy úgynevezett dilatációs (*dilatation*), majd egy eróziós (*erosion*) műveletet kell végrehajtani a képen (amennyiben a fekete a háttér és a fehér az alakzat). Ezeknél a műveleteknél a transzformált kép egy pixelét az eredeti kép egy  $B \times B$  méretű blokkja határozza meg, ahol a pixel a blokk közepében van. Az erózió esetében az új pixel érték a blokkban szereplő pixelek minimuma, azaz ha a blokkban egyetlenegy fekete pixel

szerepel, akkor a transzformált pixel is fekete lesz. Dilatáció esetén pedig az új pixel érték a blokkban lévő pixel értékek maximuma, azaz ha egyetlen fehér pixel van a blokkban, akkor az új pixelérték is fehér lesz. Maga a zárás művelete egy dilatáció és egy erózió egymás utáni végrehajtásából áll. Ennek az lesz a hatása, hogy a dilatáció eltünteti a fekete zajt a fehér alakzatból. Majd az erózió visszaállítja az alakzatot az eredeti méretre. Ez azért szükséges, mert a dilatáció alakzat méretét megnövelte. A nyitásnál minden fordítva történik a színeket tekintve, azaz a fehér zaj tűnik el a fekete háttérből. A zárást és a nyitást egymás után alkalmazva egy zajtól megszűrt képet kapunk. Ezt az egész folyamatot a 2.2-4. ábra és a 2.2-5. ábra szemlélteti.



2.2-4. ábra Zárás és nyitás folyamata.



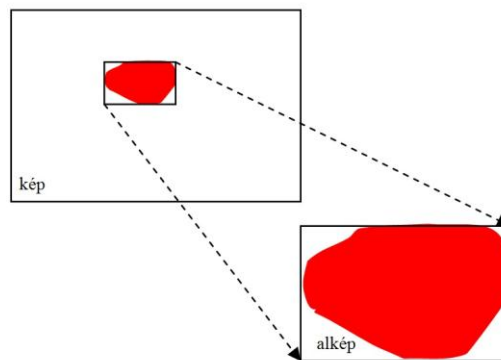
2.2-5. ábra Első képen az eredeti kép állapota, a második képen a zárás utáni állapot, a harmadik képen a nyitás utáni állapot látható. ( a kép az [5] -ös referenciából való)

### *Egyszerű mintázat azonosítás (Pattern matching)*

A mintaillesztésnél egy határozatlan elemet vizsgálunk egy már meglévő mintához képest, az egyezést keresve. A legegyszerűbb mintaillesztés egy adott érték vagy egy változók vizsgálata. A bonyolultabb minták illesztése az egyszerűbb mintákra vezethető vissza. Itt azonban nem egy adott értéket hasonlítunk össze a már elmentett mintánkkal, hanem egy csoportot. Ezeket fa mintáknak is hívjuk. Ezt a mintaillesztést használhatjuk képen történő alakzat azonosítására is. Ilyenkor egy előre megadott alakzatot (amit egy él detektáló algoritmus eltárolt) hasonlítunk össze azzal az alakzattal, amit aktuálisan dolgozott fel az algoritmusunk.

### *A lényeges terület megkeresése*

Egy adott képfeldolgozási feladatban, amikor egy képről meg kell mondanunk, hogy az tartalmazza az adott mintázatot, alakzatot vagy sem, akkor célszerű nem az egész képet megvizsgálni. Ez azt jelenti, hogy a képen végigfuttatunk egy keresést, ami az első háttértől erősen eltérő él vagy pixel után jelez, és innentől figyeljük a foltnak az  $X$  és  $Y$  irányú kiterjedését, így megkapjuk a folt pozícióját és méretét. Ezekre a kijelölt pontokra ráillesztünk egy ablakot és elmentjük egy másik képbe az ablak tartalmát. Az így kapott úgynevezett alkép (*subimage*) tartalmazza a feldolgozandó mintázatot és sokkal kisebb méretű az eredeti képnél. Ezt az eljárást *ROI(Region of Interest)* –nak nevezzük. Ezzel a módszerrel időt és erőforrást takaríthatunk meg. Az eljárás az alábbi ábrán látható.



2.2-6. ábra A teljes képből a lényeges kiemelése

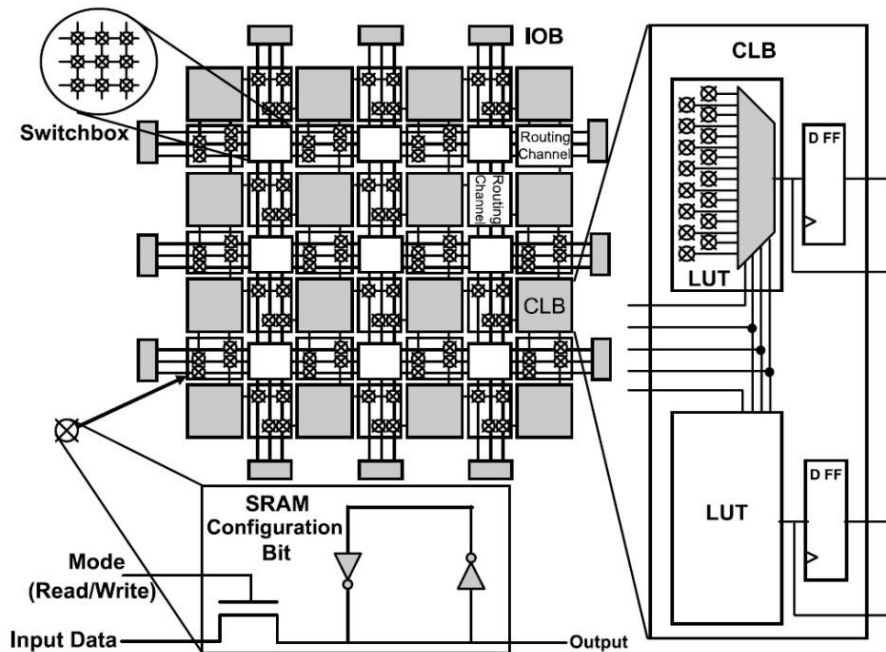
### 2.3. A robot központi egysége – az FPGA

A robot a megadott formájú szemét begyűjtéséhez alakzat felismerést használ. Mivel a robot menet közben vizsgálja meg az általa látott képen az alakzatokat, ezért az alakzatfelismerő algoritmusnak valós idejű (*real-time*) módban kell futnia, a robot vezérlése mellett. Ehhez az összetett feladathoz a robot központi számító és vezérlő egységének egy felhasználási helyén programozható logikai kapumátrixot, azaz *FPGA*-t (*Field-Programmable Gate Array*) választottunk.

A *FPGA* az általános célú processzorok és az alkalmazás-specifikus integrált áramkörök (*ASICs*) között helyezhető el. A hagyományos processzorok úgynevezett ezermesterek (*Jack-Of-All-Trades*) képesek általánosan tetszőleges kódot lefuttatni, de ez a nagy teljesítmény nagy költséggel jár. Bár a fordítók és a programozási nyelvek megváltoztatták a számítógép-tudományt, amivel lehetővé tették az emberek számára, hogy könnyen készítsenek programot számítógépre, de az általános célú processzorok által nyújtott helyenként feleslegesen nagy teljesítmény még így is nagy költséggel jár. Az alkalmazás-specifikus integrált áramkörökkel lenyűgöző teljesítményt lehet elérni, kiaknázva a párhuzamosságot egy optimalizált áramkörben. Mindez rendkívüli költségeket igénylő nagy beruházások, aminek a költségei évente növekszenek. Az *FPGA*, ellentétben az alkalmazás-specifikus áramkörökkel és a processzorokkal újrakonfigurálható és minden funkció megvalósítása hardveresen történik, úgymond vezetékesen (*hard-wired*). Ez lehetőséget kínál alacsony költséggel egy saját egyéni áramkör gyártására. Az *FPGA*-k általában lassúbbak, mint az alkalmazás-specifikus integrált áramköri megfelelőik, nem is tudnak olyan bonyolult tervezést kezelni és nagyobb az energia szükségletük is. Előnyük a rövidebb piacra kerülési időben és a helyszíni újraprogramozhatóságban és az alacsonyabb költségben rejlik. Terveznek olcsóbb, kevésbé rugalmas *FPGA* változatokat is, amelyek a tervezés lezárása után már nem módosíthatóak, azaz egyszer programozhatóak (*OTP – One Time Programmable*). Ilyen esetekben a végeredmény kerül az egyszer programozható *FPGA*-ra, maga a fejlesztés hagyományos *FPGA*-kon történik.

Noha számos fajtája létezik az *FPGA*-nak, mégis az újrakonfigurálható eszközök a leggyakoribbak. Az *FPGA* programozása és újraprogramozása logikai kapuk szintjén történik, azaz az eszközön be lehet állítani, hogy hogyan nézzen ki a logikai kapuk összekapcsolt halmaza. Egy készülő digitális áramkör vázlatát is képes emulálni az *FPGA*, azaz gyártás előtti tesztelésre is alkalmazható. Fizikailag az *FPGA* beágyazott programozható kapuk gyűjteménye egy rugalmas összekapcsolásban (2.3-1. ábra). A kapcsolásban szereplő kapuk hajtják végre a logikai függvénygenerátorokat (*LUT*), a

számítási egységeket, flip-flop-okat, az időzítést, a kapcsolható összekötő útvonalat és az I/O blokkot (IOB) a kétirányú adatátvitelre. Mivel minden logikai kapuhoz van egy megfelelő igazságtáblázat, ezért egy megfelelő áramkört lehet leképezni (*LUT*), hogy az



2.3-1. ábra Az *FPGA* összekapcsolási architektúra.

*FPGA*-n beállítsa a megfelelő igazságtáblát. Konfigurálásával a kapcsolódobozokon (*Switchbox*) beállítható, hogy melyik vezetékét hova kell csatlakoztatni. Ezt a kapcsolatot pont-pont kapcsolatként valósítja meg. Az adatok feldolgozási módját meghatározó *LUT* beállításokat és a kapcsolódobozok programozásakor keletkező adatot nevezzük konfigurációs adatfolyamnak (*bitstream*). Ezt az elkészített konfigurációt az *FPGA* számos módon tárolhatja. Az *FPGA* erre a célra használhatja például az *EPROM* / *EEPROM* vagy a *PROM* (*antifuses*) technológiákat. Bár számos architektúra használja az ilyen egyszer írható technológiát, mégis a legtöbb architektúra használja az *SRAM* programozási pontot. Az *SRAM* az *FPGA*-t felejtővé teszi, azaz a programozást minden indításnál meg kell ismételni. Egyes *FPGA*-k képesek a részleges újrakonfigurálásra is, ami azt jelenti, hogy az eszköz egyik részén folyamatosan történik a program futtatása, miközben a másik részét újraprogramozzuk.

Az *FPGA* összekapcsolásában minden logikai egység (*Configurable Logic Blocks* – *Konfigurálható Logikai Egység*) egy primitív logikai kaput valósít meg. Bonyolultabb kapuk és kapcsolások ezeknek a primitív logikai egységnek az összekapcsolásával oldható meg. Az *FPGA* felépítéséből adódóan, két összekapcsolt logikai egység nem feltétlenül kerül egymás mellé.

Általában tervezés során az *FPGA* –alapú rendszerek nagyon hasonlítanak egy hagyományos alkalmazás-specifikus integrált rendszerre. A terveket általában hardver leíró nyelven (például *VHDL* vagy *Verilog*) készítik.

A *VHDL* a *VHSIC Hardware Description Language* (*VHSIC* hardverleíró nyelv), a *VHSIC* a *Very High Speed Integrated Circuits* (nagyon nagy sebességű integrált áramkörök) rövidítése. [1]

A *VHDL* hardver leíró nyelv nem szokványos programnyelv, az ilyen nyelven írt program eredménye nem szoftver, hanem áramköri leírás, azaz egy kapcsolási rajz. A processzorok programozása során a programnyelvből fordított kód utasításait a processzor egymás után sorban végrehajtja (lefuttatja). Az áramkörleíró nyelv az áramkör egyes elemi részeinek összeköttetését definiálja, tehát nem elemi utasítások végrehajtását valósítja meg, hanem (statikus) áramköri összeköttetést definiál. [2]

Mi a feladathoz egy *Spartan 3E FPGA*-t tartalmazó fejlesztői környezetet választottunk (2.3-2. ábra). Ez a fejlesztő környezet kiváló eszköz azok számára, akik most ismerkednek az *FPGA* fejlesztéssel. A fejlesztői környezet egy *RISC* processzort és *DDR* memóriát is tartalmaz, valamint támogatja a *Xilinx Platform Flash*, *USB* és *JTAG* párhuzamos programozási interfészeket is. A panel teljes mértékben kompatibilis a *Xilinx ISE* fejlesztői környezettel, ide értve az ingyenes *WebPack*-et is.

A fejlesztői környezet több interfésszel és memóriával rendelkezik, valamint csatlókkal és LCD kijelzővel.

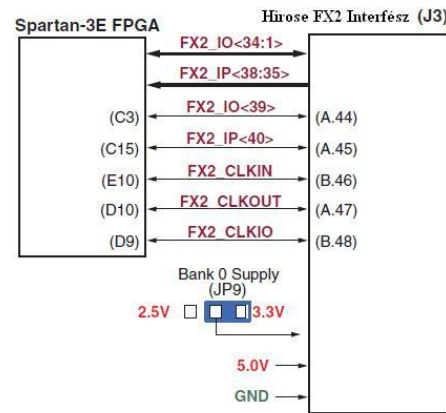


2.3-2. ábra Spartan 3E FPGA fejlesztőpanel

Csatlakozó interfészek: 100-pin Hirose FX2 interfész, Three 6-pin Pmod connectors, DB15HD VGA, PS/2 csatlakozó, Két darab DB9 RS-232 csatlakozó, RJ-45 Ethernet csatlakozó, 16-pines tűscesor az opcionális LCD-k számára, SMA csatlakozó a külső nagy sebességű órajelnek

### A Hirose FX2 interfész

Az interfész a nyák jobb oldalán található, a pontos típusa a *Hirose FX2-100P-1.27DS* (2.3-3. ábra). Ez az interfész 1,27mm-es lábtávolságot alkalmaz. Öt láb kivételével, ezek a *FX2\_IP<38:35>* és a *FX2\_IP<40>*, az összes lábon tudunk küldeni és fogadni is adatot, az említett öt láb csak bemenetként funkcionál. Ez az öt láb a csatlakoztatott bővítőkártya által generált órajel fogadására van fenntartva.



2.3-3. ábra Hirose FX2 interfész

Az *FPGA* ezen interfészen keresztül tápfeszültségeket, biztosít a csatlakoztatott eszköz számára egy +5V-os és egy elkülönített, jumper-el (*JP9*) változtatható feszültségűt. Az elkülönített ág a *FPGA's I/O Bank 0*-val (*JP9*) beállított feszültséget biztosítja, ez lehet 3,3V illetve 2,5V. Néhány *FPGA I/O* 2,5V-os feszültséget igényel, ilyen az *RSDS* és az *LVDDS*.

### Analóg digitális konverter

Az áramkör egy *Linear Technology LTC6912-1* előerősítőből és egy két csatornás *Linear Technology LTC1407A-1* ADC analóg-digitál átalakítóból épül fel. Az előerősítőt és az analóg-digitál átalakítót is sorosan programozza vagy vezérli az *FPGA*.

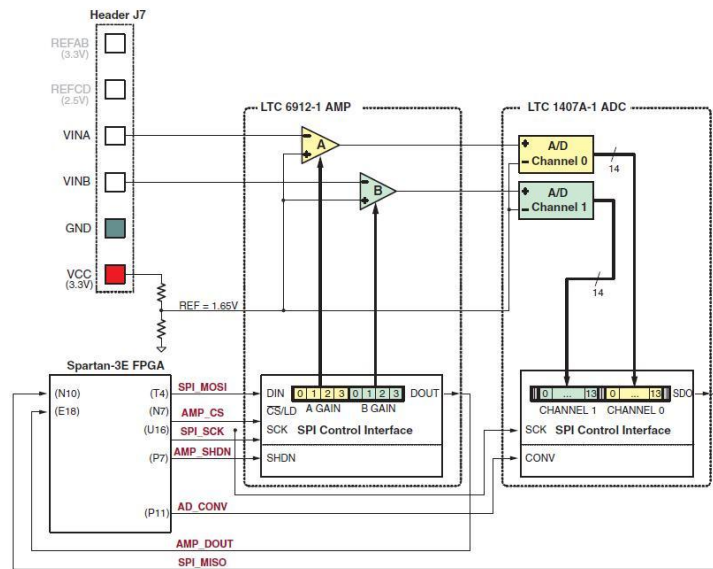
Ahogy az áramkör blokkdiagramján is látható az előerősítő kimenete az analóg-digitál átalakító bemenetére csatlakozik. Az előerősítő a *J7*-es tűkesorra érkező jelet az *SPI\_MOSI* regiszterbe írt érték szerint erősíti.

Az áramkör a *VINA* és *VINB* bemenetre érkező analóg jelet egy 14 bites digitális számmá alakítja, a következő képlet szerint:

$$D[13:0] = GAIN * \frac{VIN - 1,65V}{1,25V} * 8192 \quad (2.3.1)$$

A lehetséges erősítés értékek, illetve a maximális és minimális feszültségek a mellékletben láthatóak. Az előerősítő és az analóg-digitál átalakító referencia feszültsége

egyenként 1.65V. ezt a feszültséget egy feszültségosztó hozza létre, amely az említett ábrán látható. Az analóg-digitál átalakító +1,25V és -1,25V közötti feszültségek átalakítására képes. Az átalakítás végén kettes komplementes belső 14 bites számok jelennek meg az átalakító kimenetén. A kimeneten megjelenő 14 bites



2.3-4. ábra analóg-digitál konverter

szám minimális értéke:  $-2^{13}$ , a maximális értéke  $2^{13}-1$ .

Az LTC 6912-1 két egymástól független invertáló erősítőt tartalmaz, ezek az erősítők programozhatóak. Az erősítő vezérlő jelei a következő mellékletben láthatóak. Az SPI\_MOSI, az SPI\_MISO és az SPI\_SCK jelek megosztottak a többi, az SPI buszra kötött eszközzel.

Minden analóg bemenethez tartozik egy erősítő. A bemeneti jelek a VINA és VINB bemeneten a J7-es tükesoron érkeznek. Az erősítő az 1,65V-hoz viszonyítva erősíti azokat, az 1,65V-ot egy feszültségosztó állítja elő a J7 tükesorra csatlakoztatott 3,3V-os tápfeszültségből. Az erősítés értéke mindkét erősítőnek -1 és -100 között állítható be.

### 3. A robot komponensei

#### *A robot vázának kialakítása*

A robotváz megtervezésekor fontos volt, hogy az erős legyen, hiszen ipari környezetbe lesz használva, valamint el kell bírnia a nehéz akkumulátornak és a kézi kamerának a súlyát is. A robot vázát a *Tetrix Robotics Kit* (2.3-1. ábra) adja, ez a csomag tartalmazza a robot vázszerkezetét, kerekeket, szervo és meghajtó motorokat. A csomagban megtalálhatóak még a vezérléshez szükséges áramkörök valamint a tápellátáshoz szükséges akkumulátor is. Ezt a csomagot a *Mindstorms Lego NXT* bővítéseként olyan tanulóknak és otthoni hobbizóknak fejlesztették ki, akik szeretnének egy erősebb és nagyobb robotot építeni a *Lego NXT* csomagban található robotnál.



2.3-1. ábra: Tetrix Robotics Kit.

A könnyebb manőverező képesség, és pozicionálás elérése érdekében, a robot elején két *omni direkcionális* kerék (2.3-2. ábra) kapott helyet. Az *omni direkcionális* kerék fő jellegzetessége, hogy képes két ortogonális irányú mozgásra. Tehát az *omni direkcionális* kerékkel szerelt robot minden irányú elmozdulásra képes, az egyes kerekek összehangolt mozgatása révén. A kerék kerületén kis görgők helyezkednek el, melyek a kerék forgásirányára merőlegesen képesek elfordulni. Az *omni direkcionális* kerék használata új lehetőségeket jelent az autonóm robotok világában.



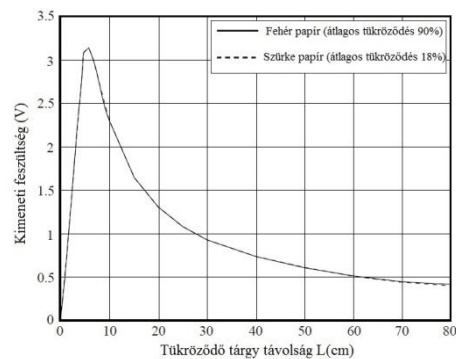
2.3-2. ábra: Omni direkcionális kerék.

A roboton elhelyezett hajtó motorok egy-egy gumírozott felületű kereket hajtanak meg egy fogaskerék áttétellel. A fogaskerék áttételt a hajtó motorok magas fordulatszáma miatt szereltük be. A robot elfordulását a motorok adott ideig tartó ellentétes irányba való forgatásával oldottuk meg. A robot elektronikáját, beleértve magát az *FPGA* alaplapot is, a *Tetrix Robotics Kit*-ben található akkumulátor látja el a működéshez szükséges energiával. A roboton két darab akkumulátort helyeztünk el, annak érdekében, ha az egyik akkumulátor lemerül a munkafolyamat közben, csak a másik akkumulátort kelljen gyorsan csatlakoztatni, valamint mert a kézi kamera felhelyezésével a robot orr-nehézzé vált, és így bizonytalanná váltak a kanyarodások.

Az akkumulátorok a robot hátulján egy erre a célra kialakított tálcán kerültek elhelyezésre. Az akkumulátorok 12V feszültséget és 900mAh-t képesek biztosítani a robot számára.

### Objektumérzékelés

Az infravörös távolság érzékelőt a robot legelejére helyeztük el egy külön az érzékelőnek kiépített tartó konzolra annak érdekében, hogy időben tudjuk az akadályokat érzékelni és elkerülni azokat. A szenzor 80cm-es távolságig érzékeli az elé helyezett tárgyakat. A tárgyak pontos távolságát a karakterisztikából kifolyólag a szenzor megközelítőleg 30cm-ig tudja megmondani, a távolság fölött a kismértékű kimeneti feszültség változás és a szenzor viszonylag nagy szórása miatt már pontosan nem adható meg a távolság. A szenzor karakterisztikája a 2.3-3. ábraán látható, ahol az y tengelyen a kimeneti feszültség (V), az x tengelyen pedig a tárgy távolsága (cm) látható. A grafikonon folytonos vonallal a fehér papíron mért adatok, míg a szaggatott vonallal a szürke papíron mért adatok láthatók. Az érzékelő által küldött jelet a fejlesztő környezet analóg digitális konvertere fogadja.



2.3-3. ábra: Az infravörös távolság érzékelő karakterisztikája.

### VDEC 1 típusú video dekóder modul

A modul (3-4. ábra) az Analog Devices ADV7183B Video Decoder chip köré épül amely képes NTSC, PAL és SECAM videojelek digitalizálására is. A modul automatikusan érzékeli azt, hogy milyen analóg jel érkezik a bemenetére, és három 54MHz-es működési frekvenciájú 10 bites A/D konverterrel digitalizálja azokat. A kimeneti adat 8 illetve 16 bites Y,Cr,Cb 4:2:2 formátumban továbbítja az FPGA felé. A modul tartalmaz minden szükséges áramkört, a stabil tápáramkört, a 75Ω-os bemeneteket, a 27MHz-es órajel generátort. A modul minden Digilent gyártmányú fejlesztői környezettel használható, ami rendelkezik a Hirose FX2 interfész-el. A csatlakoztatott modult I<sup>2</sup>C buszon keresztül kell felprogramozni, ezután a kimeneten megjelennek a digitalizált képadatok.



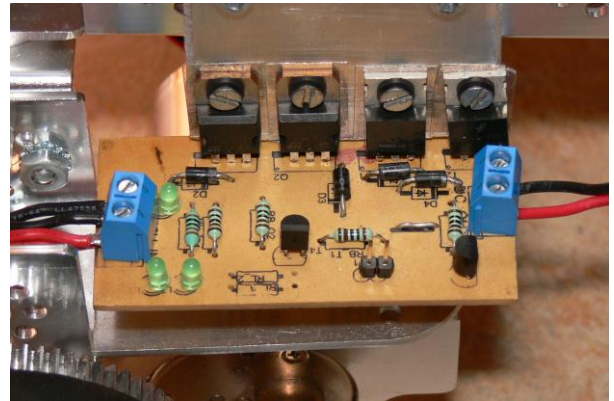
2.3-4. ábra VDEC 1 video dekóder

## A robot irányítása

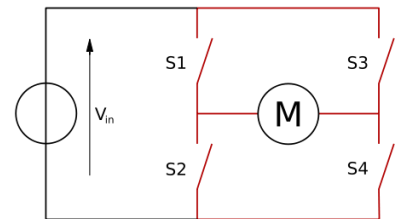
A robotot mozgató DC motorokat H-hidak (H-Bridge) segítségével vezéreljük. A H-hídvezérlést egyszerűsége mégis nagy teljesítménye miatt választottuk. Napjainkban megvásárolható integrált áramkörü formában is, mi mégis diszkrét alkatrészekből készítettük el.

A H-híd egy egyszerű egyenáramú motorvezérlő áramkör. Az áramkör segítségével a motorokat előre és hátramenetben is használhatjuk, valamint a motorok fékezése is megvalósítható. A kapcsolás legtöbbször az egyenáramú motorok előre és hátra történő forgatásához szükséges. A H-híd működése egyszerűen leírható mechanikus kapcsolók segítségével (2.3-6. ábra). A kapcsolásban négy kapcsoló található a motor tekercs körül, amelyek logikailag és működésileg párokba vannak rendezve, ezek a párok a  $S_1 - S_4$  és a  $S_3 - S_2$ . Kezdetben az összes kapcsoló nyitott állapotban van, működés közben csak egy pár lehet zárva. A  $S_1 - S_4$  kapcsolók zárásával a motoron keresztül áram folyik és a motor előre forog. A  $S_3 - S_2$  kapcsolók működtetése esetén a motor hátra felé indul meg. A motor fékezését a polaritás megfordításával végezhetjük.

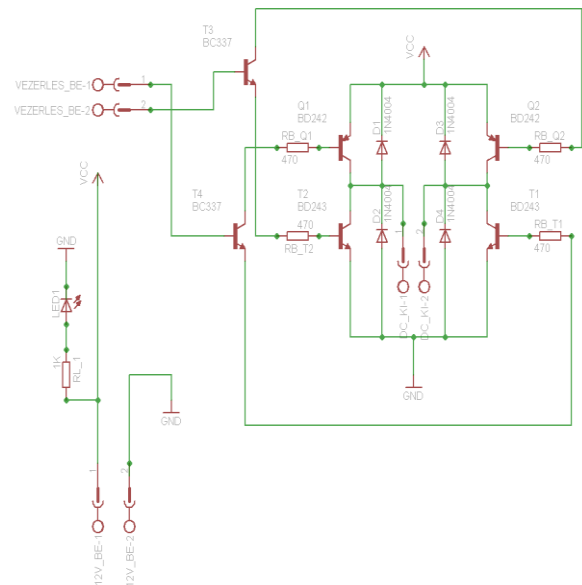
Az eredeti H-hídkapcsoláson végeztünk pár módosítást, a kapcsolásban elhelyeztünk még két vezérlő tranzisztort. Ennek a két tranzisztornak köszönhetően, az eredetileg négy vezetéken történő motorvezérlés leredukálódott két vezetékre. A kapcsolásba (2.3-7. ábra) a vezérlő tranzisztorok védelme céljából bekötöttünk még négy egyenirányító diódát is. A



2.3-5. ábra Az általunk megépített H-híd (H-Bridge)



2.3-6. ábra: A H-híd egyszerűsített rajza.

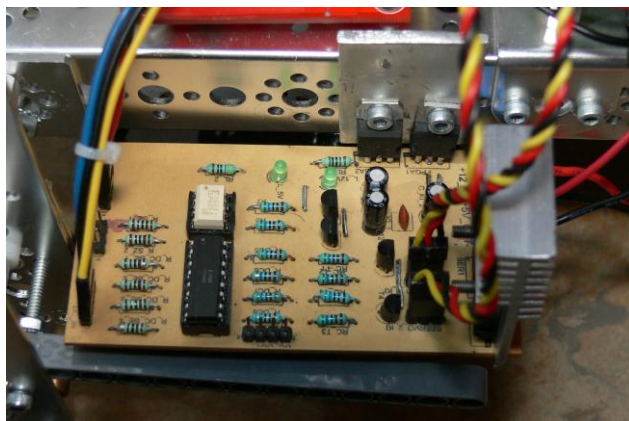


2.3-7. ábra: A H-híd motorvezérlő kapcsolási rajza

kapcsolást és a nyomtatott áramkört a *CadSoft* által fejlesztett *Eagle PCB Design* kapcsolási rajz- és nyomtatott áramkört tervező szoftver segítségével készítettük el.

### *Vezérlőpanel*

A robothoz készítettünk egy vezérlőpanelt is, amely legfőbb célja, hogy minden eszközt, ami a roboton található ellásson a hozzá szükséges árammal és feszültséggel. Ezt a feladatot feszültség stabilizátorral, és feszültség osztókkal oldottuk meg. Mivel a robot váza alumíniumból készült így ezt használtuk fel a feszültség stabilizátorok

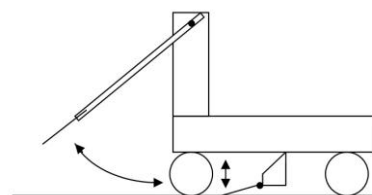


2.3-8. ábra: A vezérlőpanel.

által keletkezett hő elvezetésére. A panel második jelentős feladata az *FPGA* galvanikus leválasztása a vezérelt perifériákról, ezáltal megelőzhető, hogy az *FPGA* ki és bemenetei túlfeszültséget kapjanak egy esetlegesen hibásan működő perifériától. Ezt a galvanikus leválasztást optocsatolókkal oldottuk meg. Ezek működési elve, hogy a vezérlőjel csupán az optocsatoló belsejében lévő fotodiódát működteti, amivel szemközt egy foto tranzisztor érzékel. Így a fordított irányból érkező túlfeszültség nem tud visszajutni a vezérlő jel forrásához. Ezen a panelen keresztül vezéreljük még a *DC* motorokat, a szervó motorokat, valamint ezen a panelen keresztül oldottuk meg az infravörös távolság érzékelő illesztését az *FPGA* fejlesztő környezethez.

### *Elembegyűjtő mechanizmus*

Az alakzat összegyűjtése elsőként a megtalálásával kezdődik, melyhez a roboton elhelyezett kamerát használtuk fel, amely által szolgáltatott valós idejű képet dolgozza fel az *FPGA*. Majd a feldolgozott kép alapján eldönti, hogy az adott formát össze kell-e gyűjteni vagy nem. Ha össze kell szedni, akkor meg áll a robot és elindul az összegyűjtő algoritmus. Az algoritmus több részből tevődik össze, melyek közül az első a tároló doboz lapátjának leengedése. Ennek mozgathatóságára azért van szükség, mert amíg a robot kereső módban van, felhúzott állapotban tartjuk a lapátot, hogy



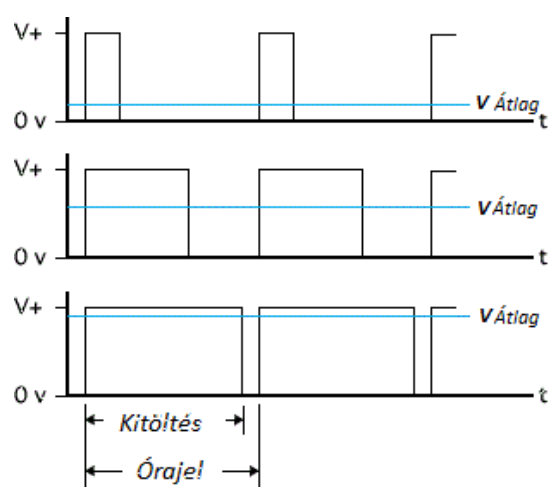
2.3-9. ábra: Az elembegyűjtő mechanizmus sematikus ábrája.

az útjába eső, de számára nem lényeges elemeket a helyükön tudja hagyni. Az összegyűjtéshez a második lépés a forma besöprése, itt a forma alakzati sajátosságai miatt nem minden esetben sikerül elsőre a besöprés így ezt a folyamatot mindig kétszer hajtjuk végre. A besöprést követően a robot felemeli, a lapátot majd folytatódik a keresés. A felhasznált tároló edényt a működéshez szükséges sajátosságok és a méretbeli követelmények miatt mi gyártottuk le.

### *Szervomotorok működtetése*

Az alakzat összegyűjtéséhez két darab szervo motor használunk. Az egyik szervo működteti a söprő mechanizmust még a másik

a dobozunk lapátját működteti. A szervók *PWM (Pulse Width Modulation)* jellel (3-10.ábra) vezérelhetők. Ennél a jeltípusnál egy órajel alatti magas értéket figyeljük. Ez a kitöltési tényező. Minél magasabb ez az érték, azaz tovább van, magas szinten a jel egy órajelen belül annál nagyobb szögben mozdul ki a szervo motor. Használunk még ilyen vezérlést *DC* motorok esetében is itt a



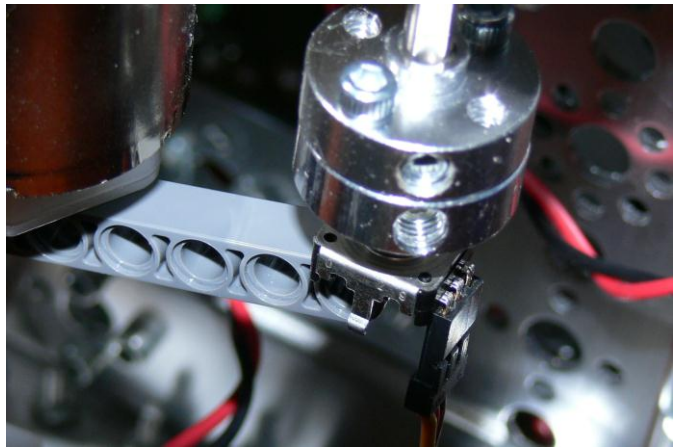
2.3-10. ábra: PWM jel különböző kitöltési tényezővel.

kitöltési tényező a motor sebességét határozza meg. A szervo motor működése alatt folyamatosan küldünk kell a szükséges *PWM* jelet, így tudjuk egy adott helyzetben megtartani a motort. A szervo motorok többsége nem fordítható teljesen körbe így nem szabad 100%-os kitöltési tényezőt kiküldeni, ugyanis ez a motor meghibásodásához vezethet.

### *Robot menetszabályzása*

Azoknál a robotoknál, amelyek nem csupán egyhelyben végzik feladatukat, egy körülményes ám mindenképpen lényeges pont a mozgatás. Ez többféleképpen is megoldható, de a legelterjedtebb a kerekek használata. Egy ilyen robotnak fordulnia is kell ezt a robot két oldalán elhelyezett két darab *DC* motorral is meg lehet oldani. Ilyenkor kanyarodásnál az egyik kerék hátra a másik pedig előre megy, így szinte egy helyben meg lehet fordítani a járművet, másik lehetőség még, hogy az egyik kereket állva hagyjuk, míg a másikat működtetjük. Így nagyobb íven fordul meg a robot, azonban az egy motoros

vezérlés minden esetben probléma mentesebb, mint a két motoros. A robot egyenes vonalú mozgásánál szintén problémákba ütközhetünk, azonban ezt nem olyan könnyű orvosolni, mint a kanyarodás esetében. Itt ugyanis szükségünk van mindkét motorra, azonban még az ugyanolyan típusú és teljesítményű motoroknak is vannak apróbb eltérései. Az eltérések részben a tekercsek feltöltődési idejéből vagy akár a csapágyak, perselyek nem



2.3-11. ábra: A kerék tengelyére szerelt enkóder

egyenlő szorulásából is adódhat. Ezek az eltérések nem észrevehetők egy olyan környezetben ahol nem szükséges a tökéletes precíz mozgítás, de egy egyenes vonalú mozgásnál ez folyamatos kitéréseket okoz és ezt folyamatosan korrigálni kell. Ennek megoldására sokféle lehetőség adódik, például digitális iránytű használata, amely az iránykoordinátákat tengelyenként 16 bit-es formában küldi ki I2C kommunikáció segítségével. Létezik elektronikus, de analóg jeleket használó iránytű is, amely tengelyenként egy-egy analóg kimenettel rendelkezik, és ezeken változtatja a feszültséget az adott pozíciótól függően. Szintén egy jó megoldás még, ha tudjuk mérni a kerekenkénti fordulatszámot. Ez a megoldás nem annyira tökéletes, mint az előbbiek, mert azt hogy a robot arra megy-e amerre mi szeretnénk, valami a robottól független dologhoz képest nézzük és az iránytű is ehhez a külső rendszerhez képest tud információkat adni a robot irányáról. A kerekek fordulatszámát figyelő lehetőség viszont csak egy visszajelzést ad számunkra, hogy a robot kerekei milyen sebességgel forognak, de ezzel a robot sebességét nem feltétlenül tudjuk meg, hiszen a kerekek csúszhatnak. Léteznek olyan motorok, amelyek rendelkeznek beépített elfordulás érzékelővel, azonban amivel mi dolgoztunk azon ilyen nem volt. Az iránytűk használatánál egyéb problémákba ütköztünk így végül az elfordulás figyelésénél maradtunk. Az ehhez szükséges eszköz egy enkóder volt (3-11. ábra). Enkóderből is létezik nagyon sokféle, de mindegyikben van egy közös, hogy magas szintű jelet ad ki legalább egyet fordulatonként. A legtöbb azonban nem csak egyet fordulatonként, hanem akár pár fokonként is egyet, így sokkal pontosabban lehet velük számolni. Egy ilyen elemet szereltünk fel a robot kerekeire. A program folyamán folyamatosan számoljuk a kerekek fordulatszámát, és ha ebben eltérést észlelünk, akkor a magasabb fordulatszámú keréknek csökkentjük a sebességét. Ezzel a megoldással sok

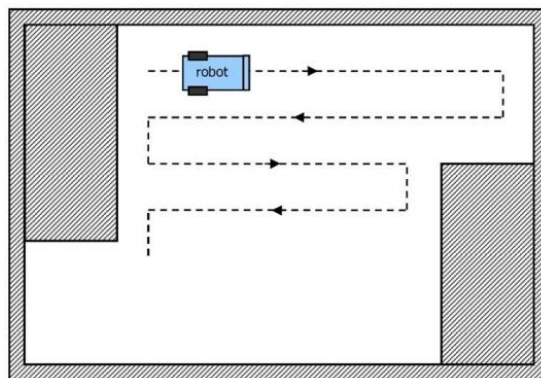
próbálkozás után sikerült egy határozott egyenes haladást elérnünk a robotnál. A kanyarodásnál nem volt ilyen egyszerű a feladat, ugyanis ott a minél tökéletesebb derékszögű fordulásra kellett törekednünk. Ehhez ki kellett számolni a körívet, amit a keréknek meg kell tennie, így a fordulatszám és a kerék külső kerülete ismeretében pontosan meg lehet határozni azt a fordulatot, amivel pontos elfordulást lehet elérni.

## 4. Az FPGA-ra fejlesztett alkalmazások

*Labview*-ban kezdtük el a robotot működtető különböző komponensek megismerését, tesztelését és felprogramozását a megfelelő feladatokra. Amikor az elkészített programrészek összeillesztésére és együttes használatára került a sor, a *Labview* nem tudta a nagy bonyolultságú programot ráfordítani a központi *FPGA*-ra. Ezt a problémát sokféleképpen próbáltuk orvosolni, de nem sikerült olyan megoldást találni, amivel folytathattuk volna a munkát a *Labview*-val, ezért el kellett hagyni a grafikus fejlesztői környezetet és egy alacsonyabb szintű, hardver közelebbi programnyelvvel a *VHDL*-lel kellett folytatnunk a munkát.

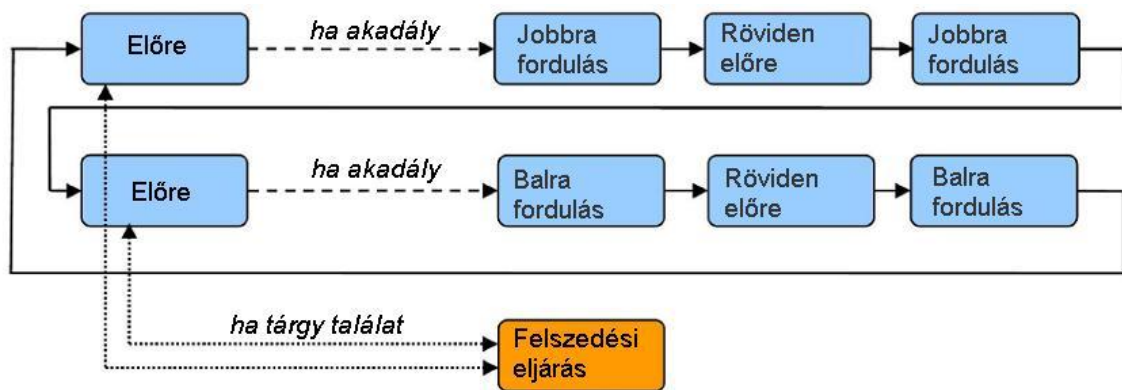
### 4.1. Terület bejárás

Egy begyűjtő robot nagyon fontos eleme az általa előre nem ismert terület minél pontosabb bejárása. Ennek megvalósítására sokféle algoritmus és lehetőség állt a rendelkezésünkre. Mi egy faltól-falig tartó területbejárást választottuk. A bejáró robot útvonala egy adott szobában az 4.1-1. ábraán látható. A robot az elejére helyezett infravörös távolság érzékelő által mért adatokat feldolgozva látja a közvetlen környezetét. Ezeknek a mért adatoknak a segítségével tudja kikerülni az útjába



4.1-1. ábra: A képen a robot területbejárása látható.

kerülő akadályokat, falakat. Ha egy tárgyhoz túl közel érkezik, végrehajt egy megforduló algoritmust. A megfordulás után saját korábbi pozíciója mellé érkezik, csak egy robotnyi széles sávval a már megvizsgált sáv mellé. Ez a módszer nem nyújt megoldást minden problémára, de egy szabályos területet elég nagy százalékban képes átvizsgálni.



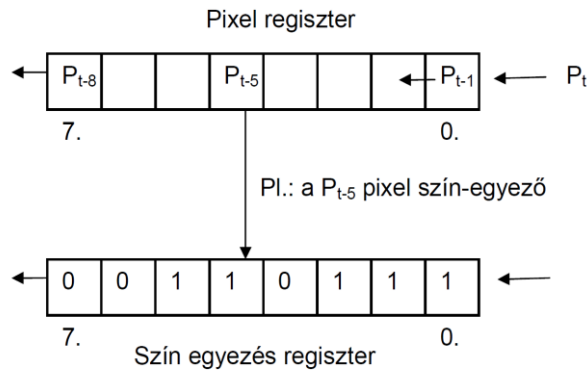
4.1-2. ábra: A képen a robot állapot átmenet diagramja látható.

A megforduló algoritmushoz egy állapotgépet (4.1-2. ábra) készítettünk, amiben a robot hét lényeges állapoton halad keresztül. Ezek közül az első az az állapot, amikor keresi azokat a földön található előre megadott típusú elemeket, amelyeket össze kell szednie. Ez az állapot mindaddig tart, amíg az infravörös távolság szenzor felől olyan jel nem érkezik, ami egy tárgy közelségét jelenti. Ilyenkor a robot megáll, majd jobbra fordul kilencven fokkal. Ezután előre halad egy rövid távolságot, majd ismét jobbra fordul kilencven fokkal, így már megtett egy száznyolcvan fokos fordulatot valamint az előzőleg átvizsgált terület mellé érkezett. Innentől ismét a hulladékkeresés következik a következő akadályig, ahol egy balra fordulás lép életbe az imént megismert módszer szerint.

## 4.2. A képfeldolgozó egység

### *Egyszerű azonosítás szín és méretek alapján*

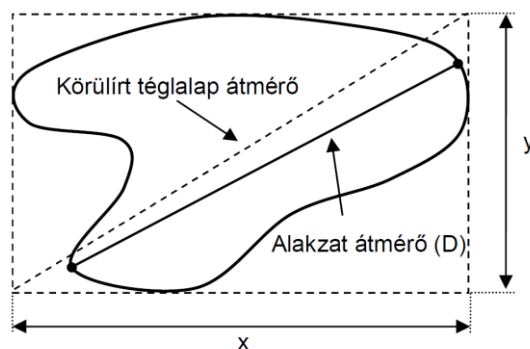
Az itt általunk javasolt objektum azonosítási technika sokkal megengedőbb, mint a későbbiekben részletezett alakzat azonosítási módszer. Itt a robot csupán az éppen elé kerülő tárgy színét és méreteit hasonlítja össze a kalibrációnál használt referencia-tárgy színével és méreteivel.



4.2-1. ábra: A pixel szín-egyezést rögzítő shift-regiszterek

Az általunk kidolgozott módszer szerint a képfeldolgozó egység a valós időben beérkező pixeleket egy 8 pixel méretű soros pixel regiszteren vezeti át, azaz a legfrissebben érkezett pixel kerül mindig a 0. indexű pozícióba és a 8 lépéssel ezelőtt érkezett pixel található a 7. pozícióban. Így a legutóbb beérkezett 8 pixelt mindig ismerjük. Minden pixelről az érkezés pillanatában megállapítjuk, hogy a referencia-tárgy színével megegyezik-e. Ezt úgy állapítjuk meg, hogy megnézzük, a kérdéses pixel szín beleesik-e a referencia szín köré szimmetrikusan felvett színtartományba (color-bin). Ha a regiszterünkben a szín-egyező pixelek száma meghalad egy beállított küszöbértéket, akkor úgy vesszük, hogy a referencia-tárgy színével egyező színű foltot találtunk. Mihelyt a regiszterben lévő szín-egyező pixelek száma a küszöb alá csökken, úgy vesszük, hogy a színfoltot elhagytuk (a soronkénti bejárás PAL szerinti rendjében.)

Ezzel a módszerrel megállapítható a színfolt legkisebb és legnagyobb x és y koordinátája a képsíkon. Ebből pedig adottak a színfolt x és y tengely-irányú méretei (lásd az ábrát).



4.2-2. ábra: Tetszőleges alakú színfolt átmérőjének és a körülírt téglalap átmérőjének viszonya

Az ábrán látható a színfolt átmérője és a körülírt téglalap, amelynek méretei x és y. Egyszerűen belátható, hogy az alakzat átmérője nem nagyobb, mint a körülírt téglalap

átmérője ( $D$ ). Másrészt a háromszögekre vonatkozó egyenlőtlenség alapján a téglalap átmérője nem nagyobb az  $x+y$  összegnél. Ez pedig egy egyszerű felső becslést jelent a  $D$  átmérőre vonatkozóan. Emellett egy szintén egyszerű alsó becslés a  $D$ -re az, hogy a  $D$  nem lehet kisebb, mint  $\max(x,y)$  (az  $x$  és  $y$  közül a nagyobb érték).

Összefoglalva, a következő igaz:

$$\max(x, y) \leq D \leq \sqrt{x^2 + y^2} \leq x + y \quad (4.2.1)$$

A képfeldolgozó egység által talált színfolt bizonyos tűréshatáron belül színegyező, az azonosításhoz tehát a méreteket kell csak megvizsgálni ennél a javasolt módszernél. A bonyodalmat az okozza, hogy a keresett forma tetszőlegesen elforgatott állapotban lehet a képen. A tárgy átmérője azonban elforgatás-invariáns paraméter, azaz a fenti egyenlőtlenségek tetszőleges elforgatás esetén is fennállnak. Ebből következik, hogy ha a talált tárgy azonos átmérőjű a referencia-tárggyal, akkor

$$x \leq x_R + y_R \quad \text{és} \quad y \leq x_R + y_R \quad . \quad (4.2.2)$$

Továbbá:

$$\max(x_R, y_R) \leq x + y \quad , \quad (4.2.3)$$

ahol  $x_R$  és  $y_R$  a referencia-tárgy színfoltjának méretei.

Ezek alapján az általunk alkalmazott kritériumok:

$$x \leq 2(x_R + y_R) \quad ; \quad y \leq 2(x_R + y_R) \quad ; \quad (4.2.4)$$

$$\frac{1}{2}x_R \leq x + y \quad ; \quad \frac{1}{2}y_R \leq x + y \quad . \quad (4.2.5)$$

Ez pontosan akkor teljesül, ha

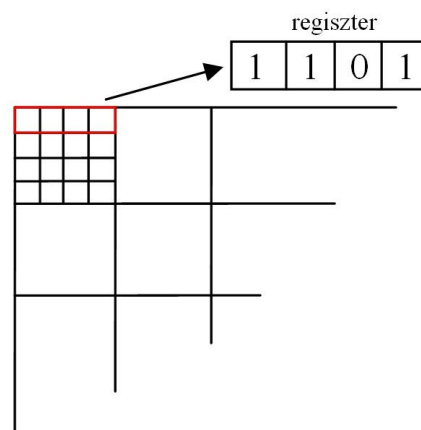
$$\frac{1}{2}D_R \leq D \leq 2D_R \quad , \quad (4.2.6)$$

ahol  $D_R$  a referencia-tárgy átmérője. Azaz a módszer pozitív választ ad minden olyan kép esetén, ahol a tárgy átmérője a referencia-tárgy átmérőjének felénél nagyobb és duplájánál kisebb.

Ez a módszer hasznos lehet olyan esetekben, amikor nem hasonló alakot keresünk, hanem bármely méret-hasonló és szín-hasonló tárgyat. Ezen kívül jó szolgálatot tehet akkor is, amikor egy gyors előzetes jelzést várunk azzal kapcsolatban, hogy a képen van-e olyan tárgy, ami további vizsgálatra érdemes

### *Felbontás csökkentés és erózió*

Az általunk fejlesztett képfeldolgozó egység egyik leglényegesebb tulajdonsága, hogy valós időben hajtja végre a képen a transzformációkat, azaz a beérkező képet nem mentjük el memóriába, mert az *FPGA* saját belső memóriája (tranzisztorokból felépített memória) nem tudja tárolni azt. Az ebben a dolgozatban javasolt módszer egyszerre hajtja végre a kívánt felbontás csökkentést és egy leegyszerűsített eróziós transzformációt (Lásd a 2.2. fejezet erózióról szóló részt.). Ennek a módszernek a másik nagy előnye, hogy egy képfeldolgozási lépés időtartama megegyezik egy videókép hardveres képfrissítési idejével a *PAL* rendszerben. Mivel a *VDEC* modul 27MHz –es frekvenciával továbbítja a pixelértékeket és három pixel érték határoz meg egy pixelt, ez azt jelenti, hogy 9MHz-enként érkezik be új pixel és a 720x288



4.2-3. ábra A felbontás csökkentés és az erózió sémája

felbontást figyelembe véve 130Hz –es frekvenciával frissül egy kép. Azaz a képfeldolgozás ideje nem több, mint  $1/130 \approx 8\text{ms}$ .

Amit tárolunk a feldolgozott pixelekből, az egy 36x36-os ( $B(x,y)$ ) márix, amely része az eredeti képnek, ami megfelel az eredeti kép egy 128x128-as részének. Ezt a felbontás csökkentést úgy érjük el, hogy az eredeti képet 4x4-es blokkokra osztjuk fel, majd a blokk egy sorát egy regiszterbe másoljuk, ahol megvizsgáljuk az egyesek számát. Ha az egyesek száma nagyobb egy megadott küszöbértéknél, akkor a sor pixel értéke egy lesz. A sorok alapján képzett nulla vagy egy értékek *ÉS* (*AND*) kapcsolatban vannak egy blokkon belül, így kapjuk meg a blokk nulla vagy egy értékét. Ez azt jelenti, hogy egy blokkon belül

akkor és csak akkor egyes az érték, ha mind a négy sor egyeseinek száma külön-külön meghaladja a küszöbértéket.

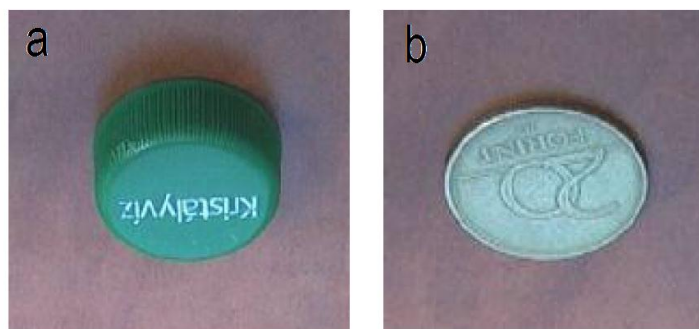
Nevezzük ezt a küszöb értéket „sűrűség-küszöbnek”, mivel ez határozza meg, hogy milyen lokális pixel sűrűség esetén lesz az aktuális blokk értéke 1 vagy 0. Ha a sűrűség-küszöb értéke 4, akkor mind a 4 pixel 1-es kell, hogy legyen mind a 4 sorban. Így egy blokkokra alkalmazott eróziós transzformációt kapunk. Ha a küszöb 4-nél kisebb, akkor azonban csak egy „részleges” eróziót” hajtunk végre. Ezt a paraméter azért szükséges bevezetni a mi esetünkben, mert a jelen esetben valós időben dolgozunk, így csak egy ilyen jellegű transzformációt hajthatunk végre. Jelen esetben ez az egy transzformáció az erózió. A paraméter bevezetésével azonban valamelyest pótolható a teljes záráshoz és nyitáshoz tartozó többi transzformáció hiánya.

#### *A sűrűség-küszöb hatásának vizsgálata*

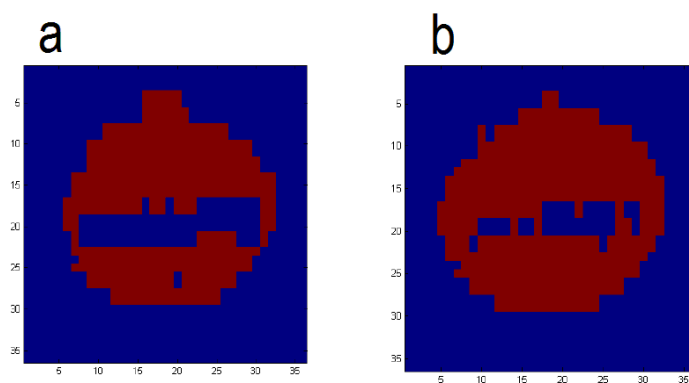
A képfeldolgozás a sűrűség küszöb 4-nél kisebb értékeire (ami a hiányzó dilatációs transzformációt hivatott pótolni), akkor alkalmazandó, amikor a tárgy színe nem homogén, bár létezik egy uralkodó tárgy-szín. Ilyen eset tesztelésére választottuk a 4.2-4. ábraán látható feliratos kupakot és a fémesen csillogó 20Ft-os érmét.

A kamera képének feldolgozását (erózió és felbontás-csökkentés) a sűrűség-küszöb 4, majd 1 értékénél végeztük el. Az eredményül kapott 36x36-os bináris mátrixokat a 4.2-5. ábra és 4.2-6. ábra képei jelenítik meg. Ahogy az várható volt, mindkét tárgy esetén a sűrűség küszöb 4 értéke mellett ritkább, az 1 érték mellett pedig tömörebb pixel mátrixot kapunk. Az írásos kupak esetében a különbség sokkal kisebb, mint a csillogó érme esetén. Ez annak köszönhető, hogy a fehér színű írás vonalvastagsága jelentősen meghaladja az erózió 4 pixeles blokk méretét, így a sűrűségi küszöb értékének csökkentése javít ugyan a kép-mátrix tömörségén, de az írás okozta hiányt eltüntetni nem tudja. A másik tárgy esetén viszont az eredmény (azaz a tömörség növelése) sokkal jobb, mivel azok a pixelek, amelyek színe a fémes csillogás miatt a kalibrált színtartományból kiesnek, sokkal szétszórtabban helyezkednek el, nem olyan koncentráltan mint a fehér színű írás esetén. A sűrűség-küszöb = 1 azt jelenti, hogy a 4x4-es blokkokon belül akár egyetlen szín-egyező pixel is 1-es blokk értéket eredményez. Így a csillogás okozta színhiba ezzel a módszerrel jól javítható. A nagyobb illetve koncentráltabb szín-eltérések azonban csak nagyobb blokk-méret alkalmazásával lennének eltüntethetőek. Ez azonban a háttérben is nem kívánt

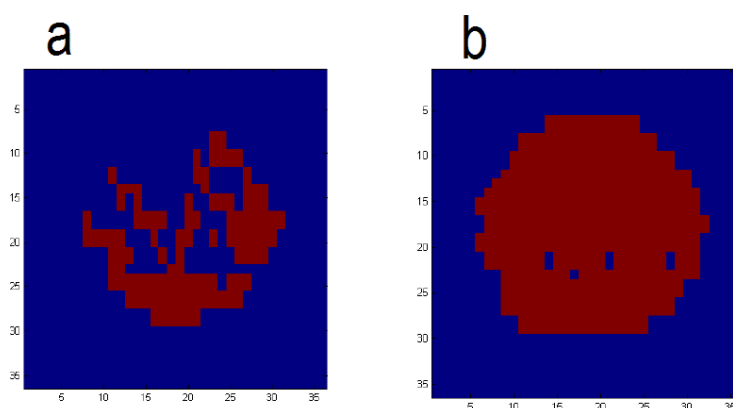
„hamis pozitív”, azaz zavaró zaj-pixelek megjelenését okozhatná a feldolgozott kép-mátrixban.



4.2-4. ábra: A sűrűség-küszöb teszteknel alkalmazott inhomogén színű tárgyak. (A kamera a felrögzítés geometriája miatt fejjel lefelé fordított képet ad a tárgyról.)



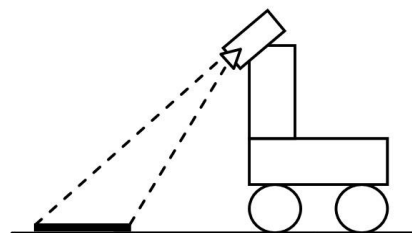
4.2-5. ábra: Az írásos kupacról készült feldolgozott kamera-kép a sűrűség-küszöb 4-es (a) és 1-es (b) értéke mellett.



4.2-6. ábra: a 20Ft-os csillagó érméről készült feldolgozott kamera-kép a sűrűség-küszöb 4 (a) és 1 (b) értéke mellett.

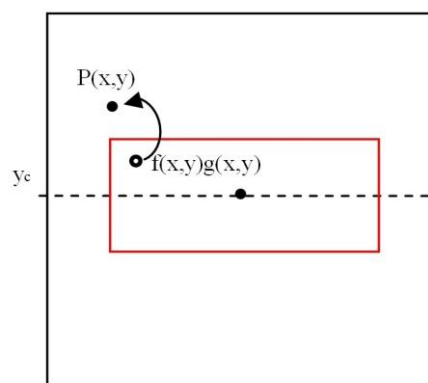
## Kép deformáció és korrekciók

A kamera állását biztonsági okok miatt nem szereltük merőlegesen lefelé néző állapotba. Ennek oka, hogy ehhez nagyon előre kellett volna helyezni a kamerát, ami ütközés veszélyt okozna. A kamerát a robot elejére biztonságos helyre szereltük, de így a kamera tengelye



4.2-7. ábra: A kamera által látott kép helyzete

nem merőleges a talaj síkjára. Ebből kifolyólag, a felvett alakzat képe deformálódik. Ennek a deformálódásnak az oka, hogy az alakzat egyik éle közelebb van a kamerához mint a másik (4.2-7 ábra). Ez kétféle jelentősebb deformációt okoz. Az egyik az, hogy a képen eredetileg egyforma  $x$  és  $y$  irányú méretek különbözőnek látszanak, azaz a kép az  $y$  tengely mentén össze van nyomva egy affin transzformációval. A másik deformáció lényege, hogy a kamerához közelebb eső tárgyak átmérője nagyobbak



4.2-8. ábra: A geometriai kép-transzformáció elve

látszik mint a távolabbiaké. A deformálódás alakzattfelismerési problémákat vethet fel, például egy eredetileg négyzet alakzat átalakul egy  $y$  tengely mentén összenyomott trapéz alakzatba. Ennek orvoslására két olyan képtranszformációt hajtunk végre amely ezeket a deformációkat helyrehozza.

Egy ilyen geometriai kép-transzformáció lényege az, hogy az új (transzformált) kép  $x$   $y$  koordinátájú pontjában a pixel értéke azonos az eredeti kép  $(f(x,y),g(x,y))$  koordinátájú pontjában lévő pixel értékkel. Azaz, a transzformációt lényegében az  $f(x,y)$  és a  $g(x,y)$  függvények határozzák meg. A transzformáció általános képlete tehát:

$$P(x, y) = P_0(f(x, y), g(x, y)), \quad (4.2.7)$$

ahol  $P_0$  az eredeti és  $P$  a transzformált kép.

A mi egyik ilyen korrekciós transzformációnk egy affin transzformáció, amely az  $y$  irányú összenyomást korrigálja. Ebben az esetben:

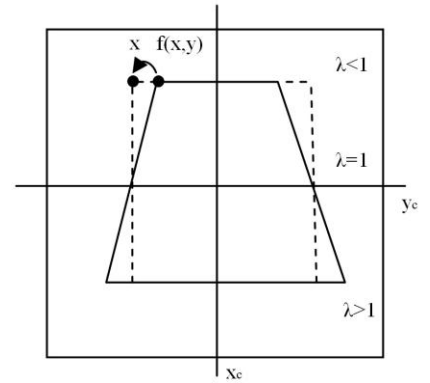
$$f(x, y) = x \quad (4.2.8)$$

$$g(x, y) = \lambda * (y - y_c) + y_c \quad (4.2.9)$$

$$P(x, y) = P_0(x, \lambda * (y - y_c) + y_c) \quad (4.2.10)$$

A  $\lambda$  értéke határozza meg az affín összenyomás illetve nyújtás értékét. A mi esetünkben  $\lambda < 1$ , azaz nyújtást alkalmazunk.

A következő egy deformáció jellegű transzformáció amely a kamerához közelebb eső tárgyak vízszintes méretét összenyomja a távolabb esőket pedig széthúzza. A kép középvonalában invariáns pontok helyezkednek el, azaz ezeket nem változtatja meg a transzformáció. Ezt egy olyan affín transzformációval adhatjuk meg, ahol  $\lambda$  értéke folytonosan változik és a kép vízszintes középvonalában pontosan 1, ahogy a mellékelt ábra mutatja. Ennek a transzformációnak a képletei:



4.2-9. ábra: Az alkalmazott korrekciós kép-deformáció

$$f(x, y) = \lambda(y)(x - x_c) + x_c \quad (4.2.11)$$

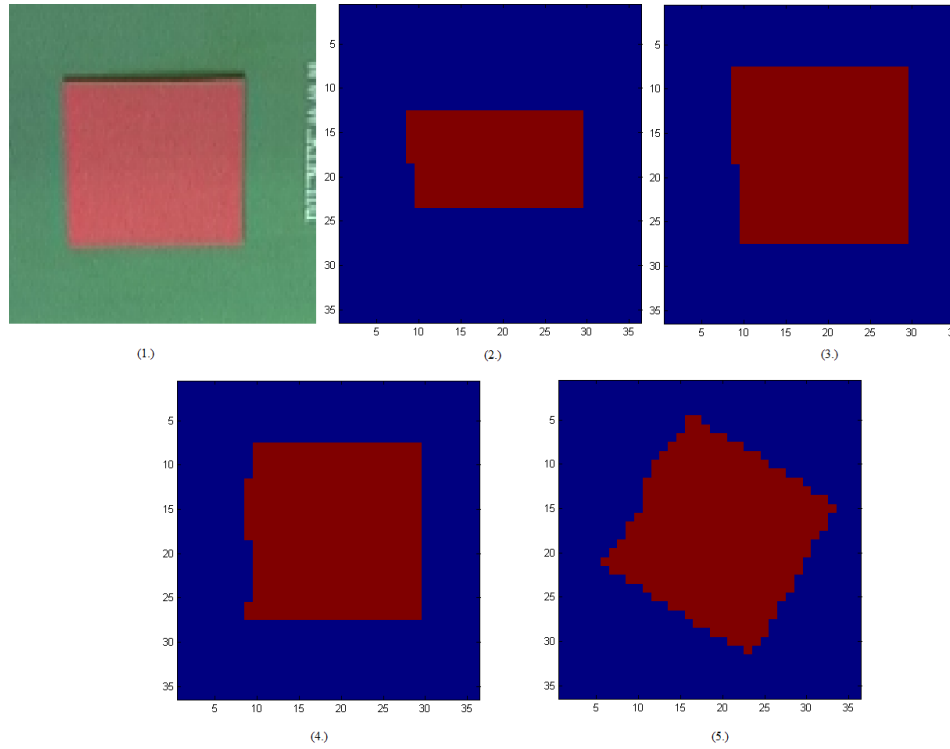
$$g(x, y) = y \quad (4.2.12)$$

$$\lambda(y) := k(y - y_c) + 1 \quad (4.2.13)$$

$$P(x, y) = P_0([k(y - y_c) + 1] * (x - x_c) + x_c, y), \quad (4.2.14)$$

ahol a  $k$  paraméter jellemzi a  $\lambda$  affín paraméter változását.

A  $\lambda$  és  $k$  értékét a két fenti transzformációnál egy kalibrációs eljárással állapítottuk meg. Ennek lényege az volt, hogy egy tökéletes négyzet alakú síkidomot felvettünk a robot kamerájával és a robot által feldolgozott torzult képen lemértük a torzulás mértékét, majd a korrekciós transzformációk paramétereit pontosan beállítottuk egy iterációs eljárással. Itt ezeket a paramétereket nem közöljük, mivel ezek az adott kamera beállításokra vonatkoznak, ha ezeket megváltoztatjuk, akkor a kalibrációt újra el kell végezni.



4.2-10. ábra: A kalibrációhoz használt négyzet, a torzult kép, a két korrekciós lépés utáni képek, végül egy elforgatott kép

Az 4.2-10 ábrán látható a kalibráláshoz használt négyzet és a kalibráció lépései. Az (1),(2),(3),(4) részában.

### *Elforgatás transzformáció*

A mintázatillesztésnél szükségünk lesz arra, hogy a képeket elforgassuk egy tetszőleges szöggel. Ennek szükségességéről részletesebben írunk az ezt követő fejezetekben. Az elforgatást is hasonló geometriai transzformációval lehet leírni, mint a fenti transzformációkat. Ennek képlete:

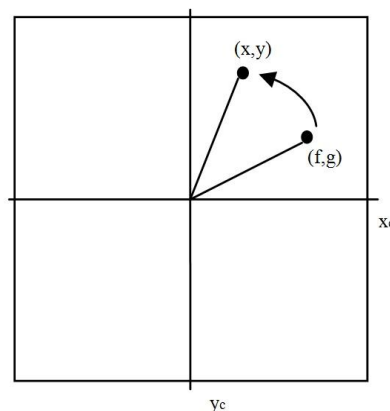
$$\underline{R} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \quad (4.2.15)$$

$$\begin{pmatrix} f \\ g \end{pmatrix} = \underline{R} * \begin{pmatrix} x \\ y \end{pmatrix} \quad (4.2.16)$$

$$P(x, y) = P_0(x \cos(\varphi) + y \sin(\varphi), -x \sin(\varphi) + y \cos(\varphi)), \quad (4.2.17)$$

ahol  $x$  és  $y$  a kép közepéhez viszonyított relatív pixel koordináták, ahogy az a 4.2-11 ábrán látható (az előző transzformációknál  $x$  és  $y$  abszolút koordináták voltak). A forgatási

transzformáció teszteléséként a korrigált négyzet mintát elforgattuk  $-30^\circ$  -kal. Az eredmény 4.2-10 (5)-ös részábráján látható.



4.2-11. ábra: A forgatási transzformáció az  $(x_c, y_c)$  kép-középponthoz, mint forgástengelyhez viszonyítva.

#### *Alakzat-felismerés mintázat illesztéssel*

Amikor a referencia-tárgy képét rögzítettük a  $36 \times 36$  pixeles bináris mintában és a kép torzításait a fent leírtak szerint kiküszöböltük, akkor készen állunk arra, hogy a folyamatosan beérkező képeken a referencia mintának megfelelő színű, méretű és alakú objektumokat keressünk. A beérkező, vizsgált képen ugyanazokat a feldolgozási lépéseket hajtjuk végre, mint a referencia-képen és egy ugyanolyan méretű, azaz  $36 \times 36$ -os mátrixban tároljuk. Ezután kezdődhet a kép vizsgálata azonosítás szempontjából. A szín és méret alapján való azonosítással foglalkoztunk már egy korábbi fejezetben. Itt a forma azonosítási módszerét ismertetjük.

A jelen munkában feltételeztük, hogy az azonosítandó tárgy:

- nem emelkedik ki jelentősen a padló síkjából,
- nagyjából síkfelületű,
- homogén színű,
- a referencia tárgyhoz hasonló alakú és
- azonos átmérőjű

Ezek alapján az egyetlen változó paraméter az, hogy az azonosítandó tárgy elforgatott állapotban van a referencia-kép helyzetéhez képest. Az előző fejezetben tárgyalt forgatási transzformációval azonban a referencia-kép tetszőleges szöggel elforgatható, csupán a helyes elforgatás szögét kell megtalálnunk, hogy a vizsgált objektummal azonos állású legyen, amennyiben az alakjuk megegyezik.

Mivel ezt a szöveget előzetesen nem ismerjük, a 0 fokostól a 360 fokosig a referencia kép minden elforgatását ki kell próbálni, valamilyen felbontással. A jelen munkában 10 fokos felbontást használtunk, azaz a  $10^\circ$ ,  $20^\circ$ ,  $30^\circ$ , ...  $350^\circ$  szögekkel forgattuk el a referencia mintát és mindegyiknél lemértük az illeszkedést a vizsgált 36x36-os felbontású képpel.

Az illeszkedés lemérésére egy egyszerű képtávolság mértéket használtunk, amit többek között az OpenCV algoritmus-könyvtár is használ [6]. Ennél az illeszkedés jellemzésére a

$$d(A, B) = \sqrt{\frac{1}{N} \sum_{\forall(i,j)} (A_{ij} - B_{ij})^2} \quad (4.2.18)$$

távolságfüggvényt használjuk, ahol az A és B mátrixok a két kép pixel értékeit adják meg és az N a pixelek száma. Ez a képlet jelentősen leegyszerűsödik azonban, ha figyelembe vesszük, hogy bináris (0 vagy 1 pixel értékű) képekkel dolgozunk, ugyanis

$$(A_{ij} - B_{ij})^2 = \begin{cases} 0 & \text{ha } A_{ij} = B_{ij} \\ 1 & \text{ha } A_{ij} \neq B_{ij} \end{cases} \quad (4.2.19)$$

Ami azt jelenti, hogy

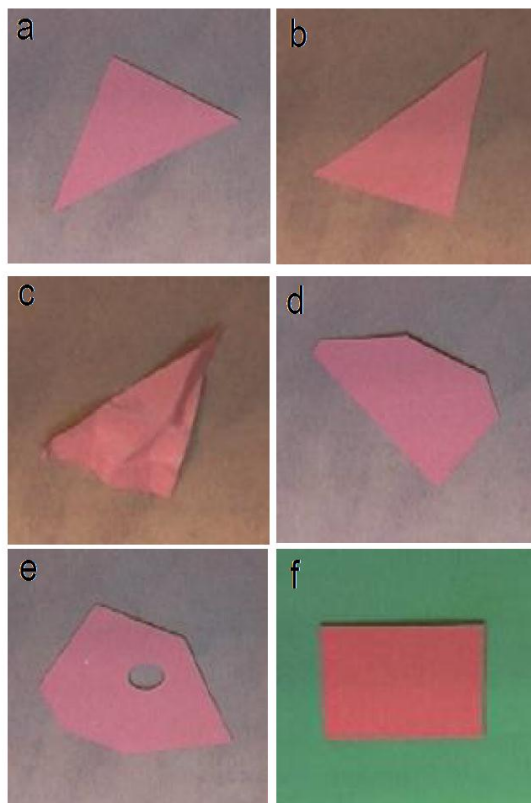
$$d(A, B) = \sqrt{\frac{1}{N} (A \text{ XOR } B)} \quad (4.2.20)$$

Ahol a XOR a mátrix elemenként végrehajtott kizáró vagy műveletet jelenti. A továbbiakban a gyökvonástól eltekintünk, mivel az illeszkedés a nélkül is mérhető, és az A XOR B mennyiségre, mint „normált képtávolságra” fogunk hivatkozni. Az illeszkedés annál jobb, minél kisebb ez a távolság.

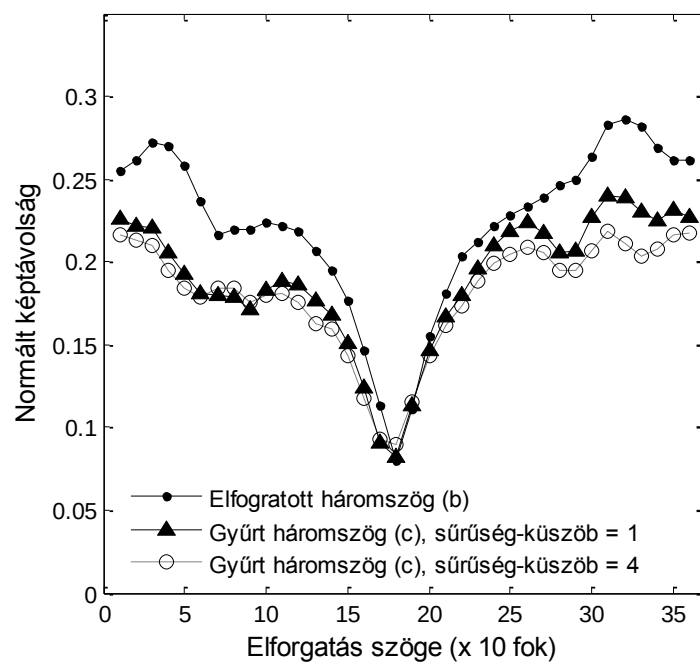
A kutatás és fejlesztés jelenlegi fázisában a blokkosított felbontás csökkentés és erózió már működőképes és tesztelt az FPGA-n, a képkorrigáló transzformációk (affin-transzformáció, deformáció és a forgatás) vizsgálata pedig úgy történik, hogy a 36x36-os méretű félig feldolgozott kép-mátrixot az FPGA soros porton kiküldi a számítógépnek egy

RS-232-es csatlakozáson át, a további kép-transzformációkat pedig *MATLAB* rendszerben fejlesztettük ki és teszteltük.

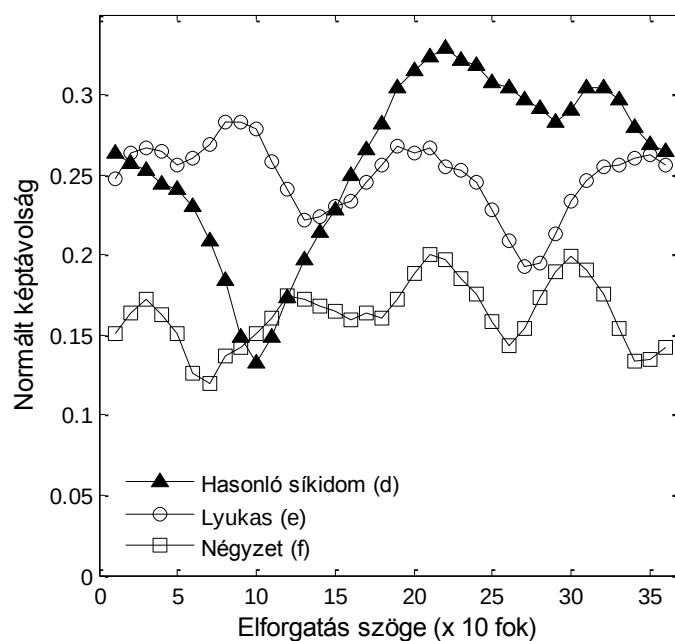
A fentiekben ismertetett azonosítási módszer tesztelésére egy egyszerű piros papír-háromszöget használtunk, amelyet a 4.2-12/a ábrán láthatunk. Ez a tárgy volt a referencia-tárgy, amelyhez hasonlót keresünk. A másik öt fotón az illeszkedési vizsgálathoz használt tárgyak láthatók: ugyanaz a háromszög elforgatott helyzetben (b), ugyanaz a háromszög gyűrt állapotban (c), egy háromszöghöz hasonló tárgy (d), egy eltérő alakú tárgy („lyukas”) (e), és a kalibrációs négyzet (f). A gyűrt háromszög feldolgozásában az eróziós lépésnél a sűrűség-küszöb 4-es és 1-es értékénél is elkészítettük a vizsgálandó képet, mivel gyűrész miatt a tárgy már nem volt homogén színű. A többi tárgynál a sűrűség-küszöb értéke 4 volt.



4.2-12. ábra: Az alkalmazott mintázat-illesztés teszteléséhez használt papír-tárgyak.



4.2-13. ábra: Az illeszkedési vizsgálat eredményei



4.2-14. ábra: Az illeszkedési vizsgálat eredményei

Az illeszkedési vizsgálat eredményeit 4.2-13 és a 4.2-14 ábrán láthatjuk. Mindegyik tárgynál a fentebb definiált relatív képtávolságot ábrázoltuk a referencia-kép elforgatási szögének függvényében. Látható, hogy az azonos illetve hasonló alakú síkidomok más jellegű görbéket eredményeznek ( a 4.2-13 görbéi illetve a 4.2-14 „Hasonló síkidom” feliratú görbéje), mint az eltérő alakú síkidomok (a 4.2-14 „Lyukas” és „Négyzet” feliratú

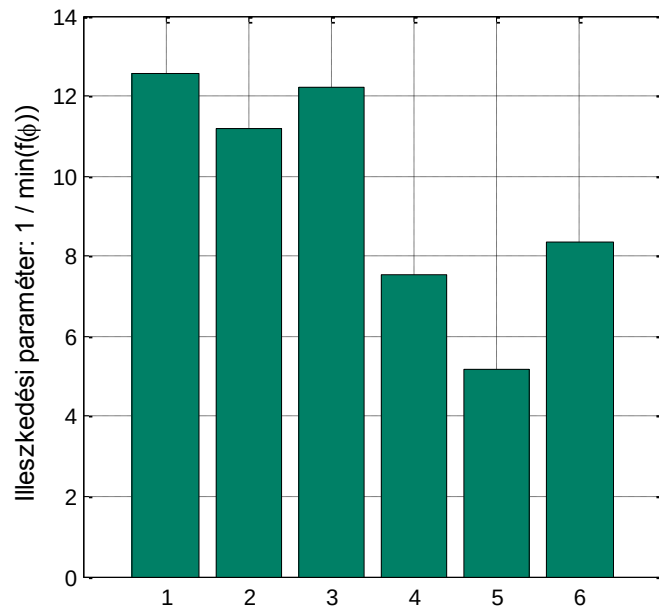
görbéje). A hasonló síkidomok határozott illeszkedési negatív csúcsot eredményeznek a referencia-tárggyal azonos szögállásnál, és a negatív csúcsnál az érték jelentősen eltér a görbe átlagértékétől. Ez az eltérő alakúaknál már nem jellemző. Az azonosítás megállapításához vezessük be a

$$p(f(\phi)) = \frac{1}{\min(f(\phi))} \quad (4.2.21)$$

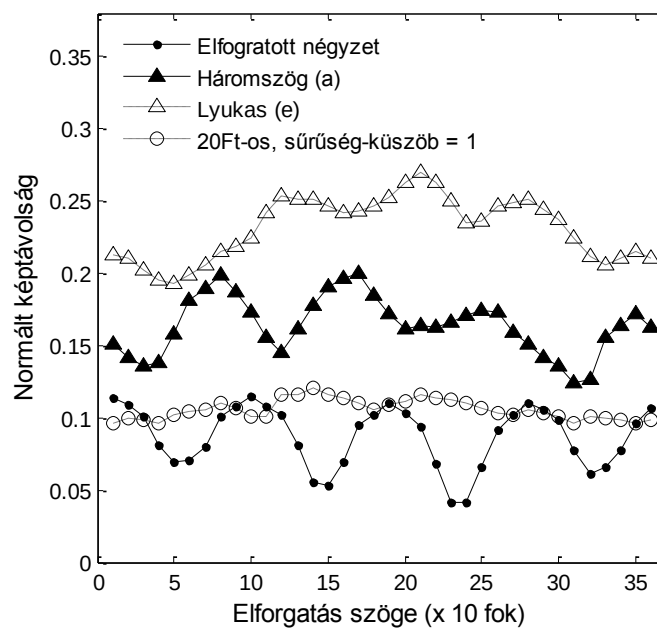
„illeszkedési paramétert”, ahol a  $\min f(\phi)$ , azaz a képtávolság görbe minimuma. Így a nagyobb érték felel meg a jobb illeszkedésnek. Ennek a paraméternek az értékeit mindegyik vizsgált tárgyra 4.2-15 ábra mutatja egy oszlopdiagramon a fotók betűzési sorrendjében. Látható, hogy az azonos vagy hasonló síkidomok által produkált értékek határozottan nagyobbak az eltérő alakú síkidomok értékeitől. Az előbbiek közelítőleg eléri a 12-es szintet vagy felette vannak, míg az utóbbiak határozottan a 8-as szint közelében vagy alatt vannak.

A fenti teszt olyan esetre vonatkozott, amikor a vizsgált tárgy csak egy kitüntetett szimmetria-iránnyal rendelkezik. Minél több szimmetria-iránnyal rendelkezik a vizsgált tárgy, annál több egyenrangú minimumot kapunk. Ennek szemléltetéseként a fent leírt tesztet elvégeztük olyan módon is, hogy a 4.2-12/f ábrán látható négyzet volt a keresett referencia-tárgy. Ennek a tárgynak négy kitüntetett szimmetria-iránya is van szemben az előző tesztnél alkalmazott háromszöggel, amelynek csak egy van. Az erre a tesztre vonatkozó képtávolság görbéket a 4.2-16 ábrán láthatjuk. Világosan látszik a négy egyenrangú minimum az illeszkedési görbéken, különösen az elforgatott négyzet esetén. A többi vizsgált teszt tárgy: ugyanaz a négyzet elforgatva, a háromszög, a lyukas síkidom és a 4.2-4 ábrán látható 20Ft-os érme a sűrűség-küszöb 1-es értékével feldolgozva. Az illeszkedési paraméter értékei ebben az esetben a 4.2-17 ábrán láthatók. Jól látszik, hogy az illeszkedési paraméterünk ebben az esetben is jól kiválasztja a referencia-tárgyat a többi közül.

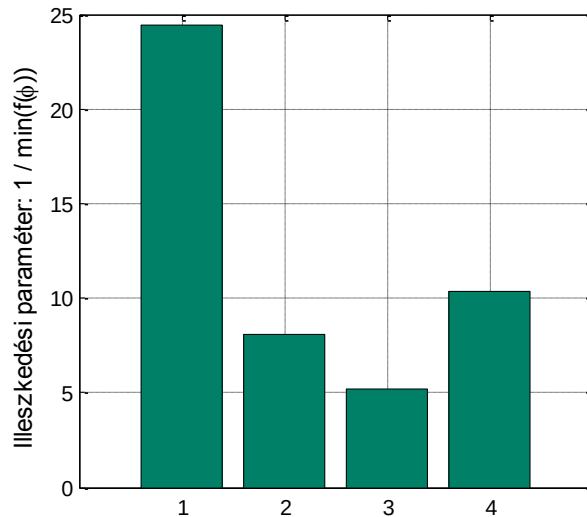
Ez a kis mintaszámra elvégzett teszt persze nem bizonyító erejű, de mindenképpen jelzi azt, hogy ez a javasolt módszer eléggé robosztus és alkalmas lehet az azonosításra illetve érdemes a továbbfejlesztésre.



4.2-15. ábra: A háromszög referencia-tárgynál mért kétféle illeszkedési paraméter értékei a vizsgált tárgyak esetében: elforgatott háromszög (1), gyűrt háromszög (2 és 3), hasonló síkidom (4), lyukas síkidom (5), négyzet (6).



4.2-16. ábra: Az illeszkedési vizsgálat eredményei négyzet esetén



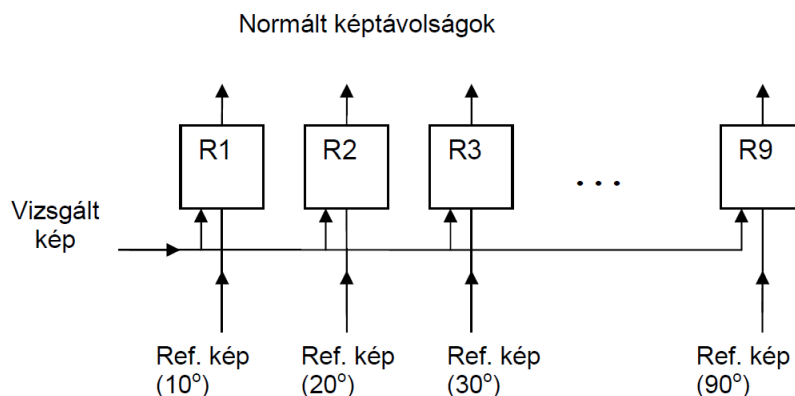
4.2-17. ábra: A négyzet referencia tárgynál mért illeszkedési paraméter értékei a vizsgált tárgyak esetében: elforgatott négyzet (1), háromszög (2), lyukas síkidom (3), 20Ft-os (4).

#### *Az alakzat-felismerési eljárás kidolgozása az FPGA-n*

Amint a bevezetőben már említettük, az FPGA nagy előnye, hogy rugalmasan alakíthatóak ki az adott célra kifejlesztett, akár párhuzamos feldolgozó áramkörök. A jelen esetben ezt a párhuzamosítást, a normál képtávolság kiszámításánál tudjuk előnyösen alkalmazni, mert az egyes elforgatott referenciaképek illeszkedés vizsgálatát egy időben több párhuzamos folyamat (processz) fogja elvégezni. Ahány referenciakép van, annyi folyamat fut egy időben. Ahogy azt a 4.-18 ábra is szemlélteti. Az R1, R2,... R9-cel jelölt processzek egy időben dolgozzák fel a 10, 20, ... 90 fokkal elforgatott referenciakép illesztését. Elég csak az első sík-negyedben vizsgálni az elforgatásokat, mivel a másik sík-negyedbeli elforgatások egyszerűen előállíthatóak szimmetria-transzformációkkal (tükrözés, 90 fokos forgatás).

Egy ilyen processz átveszi, tárolja a neki megfelelően elforgatott 36x36-os referenciaképet és a vizsgált képet, majd egy kettős ciklussal kiszámítja a normál képtávolságot. Ez 36x36x2 órajel alatt megtörténik, mivel pixelenként 2 órajel szükséges. Az egyik órajel alatt elvégezzük a XOR műveletet a két pixel-bit között, a másik órajel alatt hozzáadjuk az értéket egy összegző változóhoz. Ez azt jelenti, hogy az egész illeszkedés vizsgálat elvégezhető 2048 órajel alatt, ami kevesebb, mint egy (720 pixeles) képsor digitalizálása és továbbítása.

Így tehát ez a módszer lehetőséget nyújt arra, hogy egy rendkívül gyors előzetes tesztként használjuk, és ha illeszkedést találunk, akkor olyan időigényesebb és pontosabb vizsgálatokat is végezzünk, mint az éldetektálás vagy a kontúrkövetés.



4.2-18. ábra: A Normált képtávolság-görbe számításának párhuzamosítása az FPGA-n

## 5. A projekt jelenlegi állapota és a továbbfejlesztési tervek

A 4-es fejezet elején bemutatott egyszerű azonosítás szín és méretek alapján jelenleg tesztekkel ellenőrzötten működik a fejlesztett roboton. Tehát a robot képes egy előre kalibrált homogén színű tárgy azonosítására szín és méret alapján, továbbá képes a tárgy felszedésére és elszállítására. Emellett a robot odometria alapú navigációja szintén tesztelt és működőképes állapotban van. A forma alapján történő azonosítás tesztelése a MATLAB szoftver segítségével elkészült, ahogy azt a fentiekben részletesen ismertettük; az FPGA-n jelenleg ennek a módszernek a szimulátoros tesztelése zajlik.

A robot jelenlegi képességei lehetőséget adnak arra, hogy egy olyan ismeretlen környezetet feltérképezzen, amelyben nagyméretű a robot által érzékelhető akadályok vannak. A robot az elkészült térképet például soros porton keresztül adhatja át egy számítógépnek, ahol a térkép felbontása a robot méretével összemérhető blokkokból áll, ami körülbelül 30cm-es finomságú felbontást jelent. Az elkészült térképen a robot képes lesz megjelölni, hogy hol talált vagy esetleg látott olyan tárgyat, melyet az elején kalibráltunk. Ilyen módon egy „szennyeződés-térképet” is kaphatunk a robottól, és a szennyeződés eltávolítása lehet választható funkció is. Egy ilyen szennyeződés-térkép

felbontása viszont sokkal finomabb lehet, mint az akadályokról készült térkép, mivel a robot a terület minden pontját végigpásztázza a kamerájával és a kamera képeken a talált tárgyak koordinátáit pontosan meg tudja határozni. Így egy ilyen térkép pontosságát a navigáció pontossága határozza meg.

A jelenleg működő odometria alapú navigáció tovább javítható újabb navigációs eszközök felszerelésével. Ilyen navigációs eszköz például egy digitális iránytű, amelyről a 3. fejezetben írtunk.

## **Irodalomjegyzék**

- [1] G. Hosszú, P. Keresztes : VHDL-alapú rendszertervezés (2012) 14.o
- [2] G. Fazekas, A. Hajdu : Képfeldolgozási módszerek (2004)
- [3] J. Serra : Image Analysis and Mathematical Morphology, New York: Academic Press, (1983)
- [4] T. Huffmire, C. Irvine, T. D. Nguyen, T. Levin, R. Kastner, T. Sherwood : Handbook of FPGA Design Security (2010)
- [5] I. Engedy, G. Horváth : A Global, Camera-based Mobile Robot Localization (2009)
- [6] G. Bratski, A. Kaehler : Learning OpenCV: Computer Vision with OpenCV Library, O'Reilly (2008)
- [7] J. Canny, "A computational approach to edge detection," IEEE Transactions on Pattern analysis and Machine Intelligence 8 (1986): 679-714.
- [8] S. Suzuki and K. Abe, "Topological structural analysis of digital binary images by order following," Computer Vision, graphics and Image Processing 30 (1985): 32-46.
- [9] J. Semple and G. Kneebone, Algebraic Projective Geometry, Oxford, UK: Oxford University Press, (1979)