

Gépi látás - képfeldolgozás és mélységi kép két webkamera segítségével



Szerző:

Szabó Márk

Konzulens:

Dr. Aradi Petra, BME-MOGI

2013.

Tudományos Diákköri Konferencia

Budapesti Műszaki és Gazdaságtudományi Egyetem

Tartalom

1.	Bevezetés	3
2.	Az NI LabVIEW-ról általában	4
2.1.	Bevezetés	4
2.2.	Az adatfolyam programozás alapjai	4
2.3.	Változók.....	5
2.4.	Cluster-ek.....	6
2.5.	Ciklusok.....	7
2.6.	Express VI-ok	8
3.	Egyszerű képfeldolgozási technológiák.....	8
3.1.	Szükséges modulok	8
3.2.	Kamera beállítása Express VI segítségével	9
3.3.	Kamera beállítása klasszikus VI-ok segítségével	10
3.4.	Szín alapú egyezés keresés express VI-ok segítségével	12
3.5.	Szín alapú egyezéskeresés sima VI-ok segítségével	15
3.6.	Forma alapú egyezéskeresés express VI-ok segítségével.....	17
3.7.	Forma alapú egyezéskeresés sima VI-ok segítségével	20
	Kiegészítés	22
4.	Magasabb szintű képfeldolgozási technikák.....	24
4.1.	Kétdimenziós vonalkód olvasás express VI-ok segítségével	24
4.2.	Kétdimenziós vonalkód olvasás sima VI-ok segítségével.....	25
5.	Mélyégi kép.....	26
5.1.	Bevezetés	26
5.2.	A sztereo rendszer kalibrálása	27
	Használat közben és a Front Panel.....	27
	A program működése, a Block Diagram.....	30
5.3.	A sztereo rendszer használata.....	32
	Használat közben és a Front Panel	32
	A program működése és a Blokk Diagram	33
6.	Források, felhasznált irodalom	35

A dolgozat lezárásának időpontja: 2013.11.08.

1. Bevezetés

Dolgozatom a gépi látás megvalósításáról szól LabVIEW környezetben. Változatos képfeldolgozási technológiákat mutatok be folyamatosan szem előtt tartva, hogy a módszerek szűkebb költségvetésű egyetemi és hallgatói projektekben is használhatóak legyenek.

Először áttekintem a LabVIEW programozás alapjait, az adatfolyam programozás mibenlétét, előnyeit, hátrányait. Csak olyan mélységig, hogy a későbbi fejezetek azok számára is érhetőek legyenek, akik még nem programoztak LabVIEW környezetben.

Ezután rátérek az egyszerű képfeldolgozási módszerekre. Itt először a színalapú mintázatkeresést mutatom be. Ennek keretében a megadott színbeli mintázatot keresi a program a képen. Majd a fekete-fehér képen történő, forma alapú mintázatkeresést írom le. Itt a különböző módszerekkel monokrómmá tett képen keresi az előre definiált alakzatot.

Majd folytatatom a magasabb szintű képfeldolgozási technológiákkal. Itt a kétdimenziós vonalkód (QR kód és Data Matrix) olvasásáról írok, melyek egyre elterjedtebbek mind az iparban, mind a mindennapi életben.

Ezt követően bemutatom azt, hogy két egymás melletti, csupán horizontálisan eltolts webkamera képét felhasználva hogyan lehet - az emberi látáshoz hasonlóan - mélységi képet előállítani. Itt fontos a megfelelő beállítás, ami a kameráknak több szögből megmutatott fekete-fehér ráccsal végezhető. Ezután a beállítás természetesen elmenthető, így ezt elegendő kamera-beállításonként egyszer elvégezni. Ezt követően a program képes a webkamerák képét élőben (real time) mélységi képpé alakítani és azt egy színes grafikonon megjeleníteni. Az összes ponthoz rögtön rendel egy mélységi koordinátát is, amit egyrészt a színekkel érzékeltet, másrészt leprogramoztam azt is, hogy ha az egeret az adott pont fölé visszük, akkor külön is megjeleníti a mért távolságot. Természetesen ennek a technológiának is megvannak a korlátai, nem várható el, hogy két középkategóriás webkamera képéből tökéletes háromdimenziós képet kapjunk, de az meghatározható belőle, hogy a kamerák előtt mely területen van közeli tárgy, aktivitás, amin a korábban ismertetett képfeldolgozó technológiákat felhasználni érdemes. Ezzel leszűkíthető a vizsgálat területe, amivel erőforrást takaríthatunk meg, valamint egy mozgásra képes robot részeként használva a módszert, a robot képes lesz az adott aktivitás felé fordulni. Emellett a módszer mérési tartománya nagyban függ a kamerák elrendezésétől. Egymáshoz nagyon közel elhelyezett kamerák esetén elsősorban közelebbi célpontok esetén fog pontosabban működni, míg távolabb elhelyezettek elsősorban távolabbi célpontoknál adnak használhatóbb eredményt.

Minden bemutatott technológia esetén kitérek a megvalósítási lehetőségek széles skálájára, illetve az adott technológiában rejlő lehetőségekre és korlátokra is, amelyek egyrészt az adott módszer fizikájából származnak, másrészt a felhasznált kamerák hibája. Ez utóbbi azonban szándékosan része a dolgozatomnak, hiszen célom nem az volt, hogy ipari képfeldolgozó technológiákról írjak, hanem pont ezeknek az egyszerűbb eszközökkel való kivitelezését mutassam be, így a hallgatók számára is felhasználhatóvá tegyem őket.

2. Az NI LabVIEW-ről általában

2.1. Bevezetés

Ebben a fejezetben röviden áttekintem a LabVIEW programozás alapjait. Célom nem egy teljes értékű LabVIEW tanfolyam (nem is lehetne ilyen röviden leírni), hanem csupán annyi, hogy azok számára is érthetőek legyenek a későbbi fejezetek, akik még soha nem használták ezt a fejlesztőkörnyezetet, azonban más nyelvben már rendelkeznek programozási tapasztalatokkal.

A LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) a National Instruments fejlesztői környezete a szintén általuk létrehozott grafikus programnyelvhez. Ennek a programnyelvnek a neve „G”, ami nem összetévesztendő a G-kóddal, amiben CNC gépeket programozhatunk.

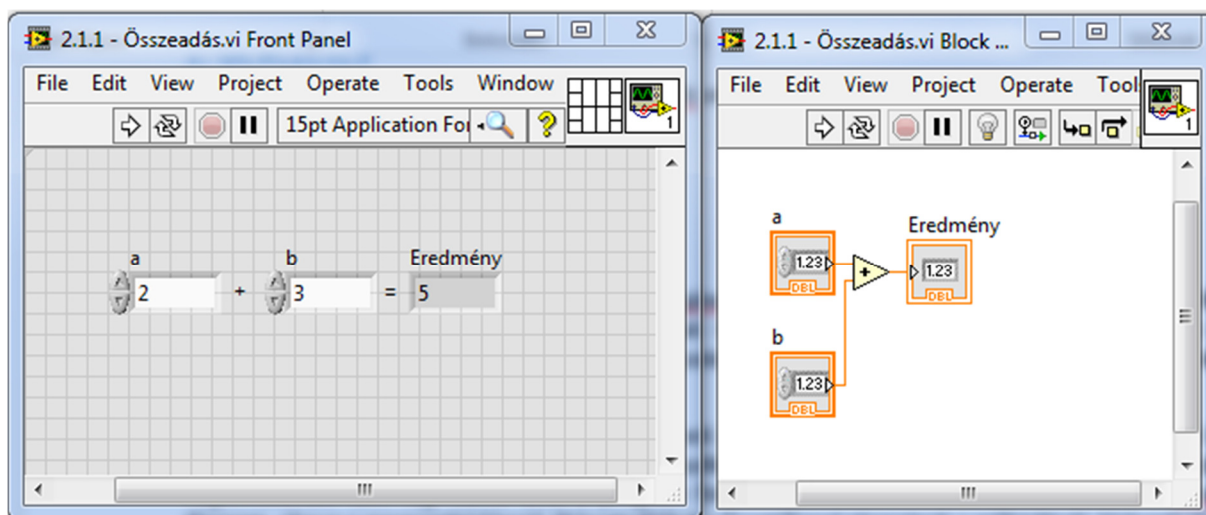
2.2. Az adatfolyam programozás alapjai

A „G” nyelv az adatfolyam programozás elvét követi. Egy LabVIEW program alapvetően két részből áll: a grafikus elemeket (gombokat, szövegmezőket, grafikonokat stb.) megjelenítő Front Panel (előlap), amit majd a végfelhasználó látni fog, illetve amit a program futása közben mi is használunk, valamint a Block Diagram, ahova a programkód kerül. Itt kontrolláljuk a Front Panel elemeinek működését, kiolvashatjuk tartalmukat, módosíthatjuk tulajdonságaikat, kezelhetjük eseményeiket.

A LabVIEW programkód erősen eltér a szokásos programnyelvekben írt kódtól. Itt nem parancsokat kell gépelnünk, hanem ehelyett a jobb egérgombra megjelenő palettából (Functions menü) hívhatunk elő elemeket, amiket kattintással tudunk elhelyezni a Block diagramon. A Front Panelen a hasonlóan kezelhető Controls palettából választhatjuk ki a grafikus felhasználói felület elemeit. Az általunk létrehozott programok neve Virtual Instrument, röviden VI, magyarul virtuális műszer, utalva a LabVIEW fejlesztés kezdetétől használt jelmondatra, „The software is the Instrument” (a szoftver a műszer). Ennek megfelelően programjaink kiterjesztése *.vi. Azonban nem csak névben egyeznek meg a beépített elemek és az általunk

írt programok, hanem a saját programjain is ugyanúgy beszúrhatók egy másik programba, mint a beépített VI-ok. Az általunk írt, más (esetleg több) programban felhasználásra szánt programrészeketszokás subVI-oknak is nevezni.

A Front Panelen elhelyezett Control (bemenet) típusú elemek értékét futás előtt vagy közben mi állíthatjuk, a beállított értéket a Block Diagramon lévő programkód azonnal eléri. A szintén előlapon lévő Indicator (kimenet) jellegű elemek értékét a programkód állítja be. A Block Diagramon elhelyezett, a LabVIEW beépített elemeit, saját subVI-ainkat vagy mások által készített blokkokat használó programkódunk végzi el a szükséges műveleteket a program változóin. A bemeneteket, kimeneteket és a programkód elemeit figyelhetjük meg a következő egyszerű programon.

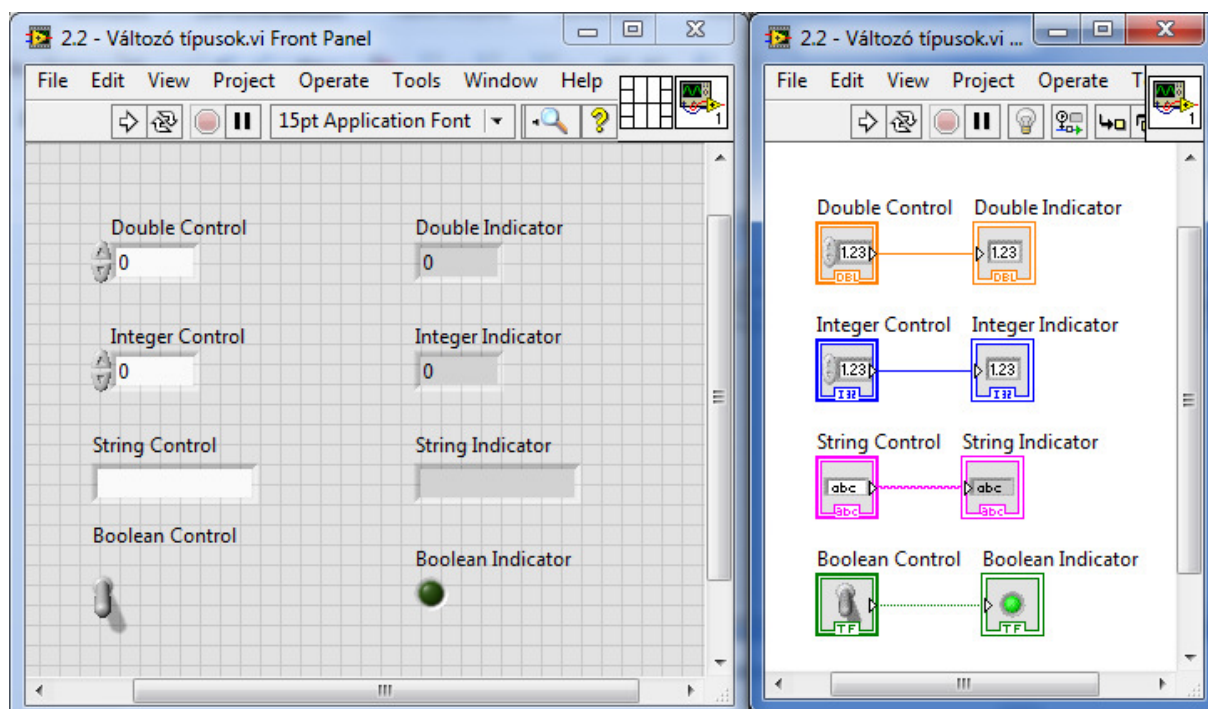


2.2.1. ábra – Összeadás

A Front Panelen beállítjuk a és b értékét, majd a  (Run) gombra kattintva indíthatjuk a programot. Ekkor a Block Diagramon lévő program kiolvassa a és b értékét, átadja az összeadó VI-nak, ami visszaadja az összegüket, amit megjelenítünk az *Eredmény* indikátorban.

2.3. Változók

Láthatjuk, ahogy a változók értéke a színes „vezetékeken” halad egyik bloktól a másikig. A „vezetékek” színe mindig reprezentálja a változó típusát. A következő mintaprogram tartalmazza az alapvető változókat:

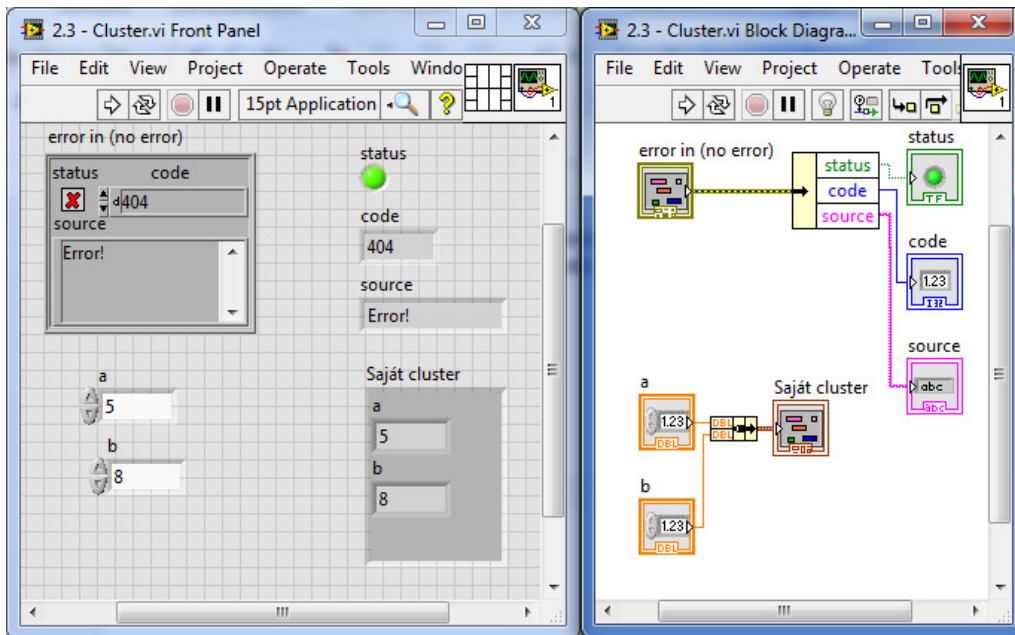


2.3.1. ábra – Változó típusok

Mindegyik változóból csinálhatunk tömböt, ekkor vastagabb lesz a vonal. Csinálhatunk több dimenziós tömböt is, a vonal ekkor még tovább vastagodik.

2.4. Cluster-ek

A klasszikus programnyelvekben a rekordok állnak a legközelebb a cluster-ekhez. Röviden arra jók, hogy több különféle típusú változót tudunk összekapcsolni és egy adatfolyamként kezelni. Ezáltal átláthatóbbá válik a program, főleg akkor, ha két subVI több információt is szeretne átadni egymásnak. Jellegzetes beépített elem az error cluster, ami tartalmaz egy boolean (igaz/hamis) változót, hogy történt-e hiba, egy integer-t (egész számot), a hiba kódját és a hiba leírását string-ként. Emellett természetesen mi magunk is létrehozhatunk clustereket tetszőleges tartalommal.

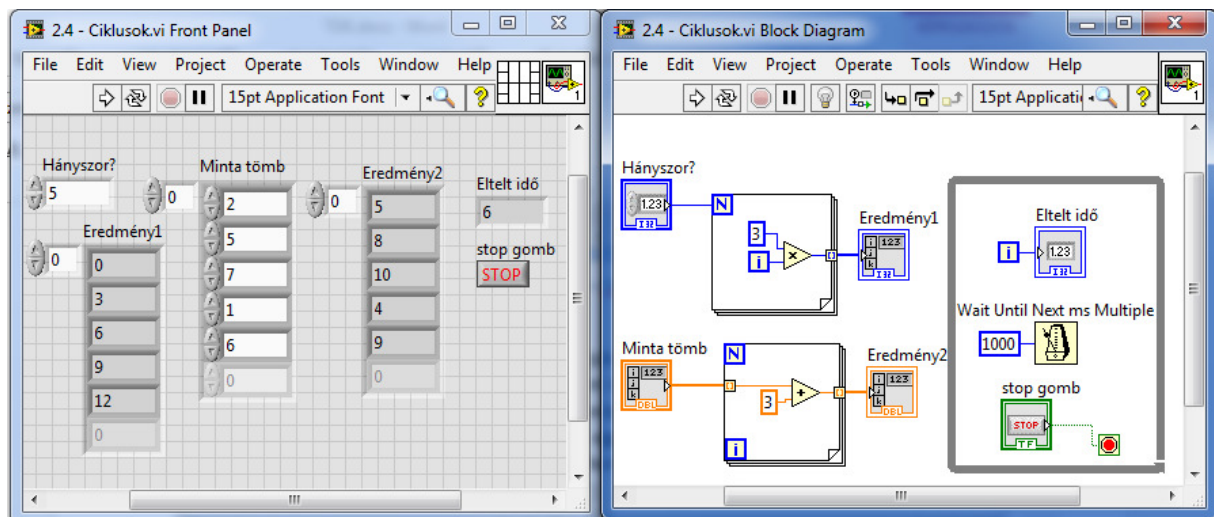


2.4.1. ábra – Cluster-ek

2.5. Ciklusok

A többi programnyelvhez hasonlóan itt is elérhetjük a klasszikus for, while és switch struktúrákat. LabVIEW-ban különlegesség, hogy ha egy for ciklusra bemenetként rákötünk egy tömböt, akkor alapértelmezettként addig fog futni a ciklus, amíg végig nem megy a tömb összes elemén, miközben a ciklusból mindig az aktuális elemet érhetjük el. Ha a for ciklus belsejéből a ciklus keretére kimenetként kötünk egy skalár változót, akkor ott alapértelmezettként létrejön belőle egy tömb és abba tárolja szép sorban a létrejött változókat. Természetesen mindkét indexelési beállítás átállítható.

A következő példában láthatjuk mindkét viselkedést. Emellett egy while ciklus addig fut, amíg le nem nyomjuk a stop gombot. A számolást a Wait Until Next ms Multiple VI segítségével hajtja végre. Ez a VI a ciklus végeztével annyit vár, hogy összesen a rá bemenetként kötött (esetünkben 1000) milliszekundum ideig tartson egy ciklusfutás. Így mindegyik kör pont 1 mp lesz, vagyis a ciklus sorszáma (az i) éppen az eltelt másodpercek számát adja.



2.5.1. ábra - Ciklusok

2.6. Express VI-ok

A képfeldolgozás során is gyakran használunk Express VI-okat, így muszáj róluk is pár szót ejteni. Első ránézésre úgy néznek ki, mint egy rendes VI, csak amikor lerakjuk őket a Block Diagramon, akkor rögtön megnyílik egy varázsló, ahol konfigurálhatjuk a VI-t. Erre jó példa a képfeldolgozás során szinte mindig használt Vision Acquisition, amivel a használt kamerát állíthatjuk be. Ennek részletes leírására a következő fejezetben térek majd ki.

3. Egyszerű képfeldolgozási technológiák

3.1. Szükséges modulok

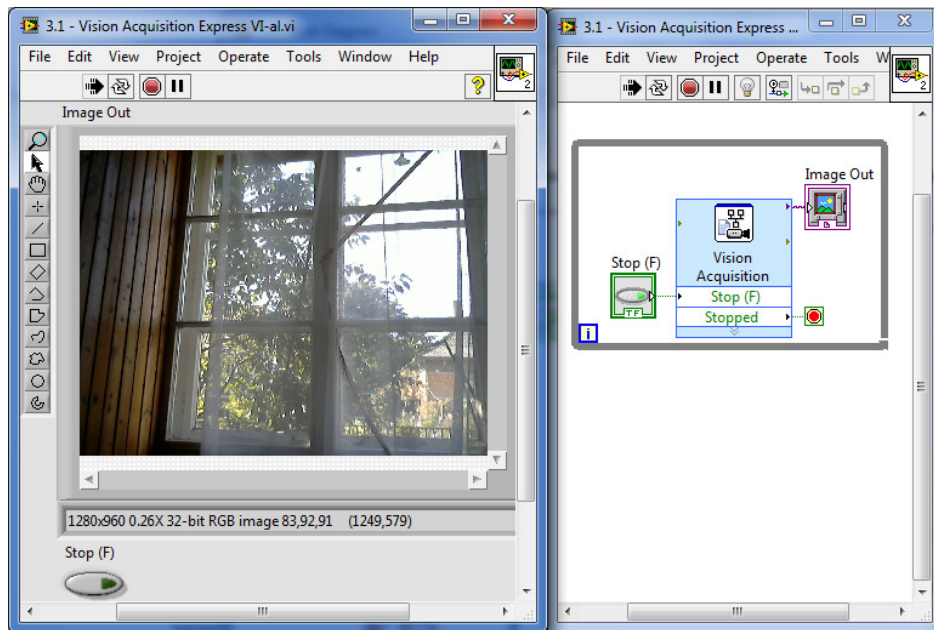
A képfeldolgozáshoz egyrészt szükségünk lesz a LabVIEW valamelyik verziójára. Én 2013-asat használok dolgozatom készítése során. Ezen kívül szükségünk lesz a következő modulokra:

- NI-IMAQ 4.6.4
- NI Vision Acquisition Software 2013.08.
- NI Vision Development Module 2013.
- NI Vision Builder for Automated Inspection 2012. SP1

A feltüntetett verziószámok az általam használt verziókat jelölik. Valószínűleg korábbi és későbbi verziókkal is működnek a későbbiekben leírtak, bár ott lehetnek eltérések. Az összes program próbaverziója letölthető a National Instruments oldaláról (<http://www.ni.com>).

3.2. Kamera beállítása Express VI segítségével

Ahhoz, hogy bármilyen képfeldolgozási technológiát használni tudjunk, először is hozzá kell férnünk a kameránk képéhez. Ehhez az előző fejezet végén említett Vision Acquisition Express VI-t használom. Ezt a VI-t a Functions menüben (jobb gomb egy üres területen a Block Diagramon) találjuk a Vision and Motions > Vision Express almenüben. (Ha nem találjuk, használjuk a keresést, ha úgy sem, akkor ellenőrizzük, hogy az összes szükséges kiegészítő telepítve van-e.) Miután letettük a VI-t, rövid várakozás után megnyílik egy varázsló. Az első ablakban az NI-IMAQdx Devices alatt választhatjuk ki a kameránkat. Ha ez megvan, kattintsunk a tovább (Next) gombra. A következő lapon választhatjuk ki a mintavételezés típusát. Válasszuk a második opciót (Continuous Acquisition with inline processing). Így képesek leszünk valós időben feldolgozni a kamera képét. A következő lapon választhatjuk ki a kamera videójának minőségét és egyéb tulajdonságait (kontraszt, féhéregyensúly stb.). Alapértelmezettként a legjobb minőséget és mindenből az automatikusát választja ki a rendszer. Általában hagyhatjuk is ezt a beállítást, később igény szerint módosíthatjuk. A következő lapon állíthatjuk be a képek naplózását (Image logging), vagyis a kamera képének merevlemezre mentését. Ez lelassítja a mintavételezést, így csak szükséges esetben kapcsoljuk be. Az utolsó lapon állíthatjuk be az express VI be- és kimeneteit. Bemenetként bekapcsolhatjuk ugyanazokat, amiket a 3. lapon beállíthatunk (pl. a fényességet igen, de a felbontást nem). Így a program futása közben is módosíthatjuk ezeket a tulajdonságokat, akár a felhasználó interakciója alapján is. Ugyanígy beállíthatunk kimeneteket is, bár csak a kép számát (Image Number) és a képfrissítés gyakoriságát (Frame Rate). Ezután már nyugodtan kattinthatunk a Finish-re és a beállított Express VI rövid várakozás után megjelenik a Block Diagramunkon. A második lapon beállított tulajdonságoknak megfelelően rögtön létre is jön egy while ciklus egy stop gombbal és egy Front Paneli megjelenítővel. Hasznos, ha erre a kijelzőre még most jobb gombbal rákattintunk és kiválasztjuk a Zoom to Fit opciót. Így eltűnnek a görgetősávok és a kamera képét akkorára kicsinyíti, hogy beférjen a megjelenítőbe. Most így néz ki a programunk:



3.2.1. ábra- Vision Acquisition Express VI-al

3.3. Kamera beállítása klasszikus VI-ok segítségével

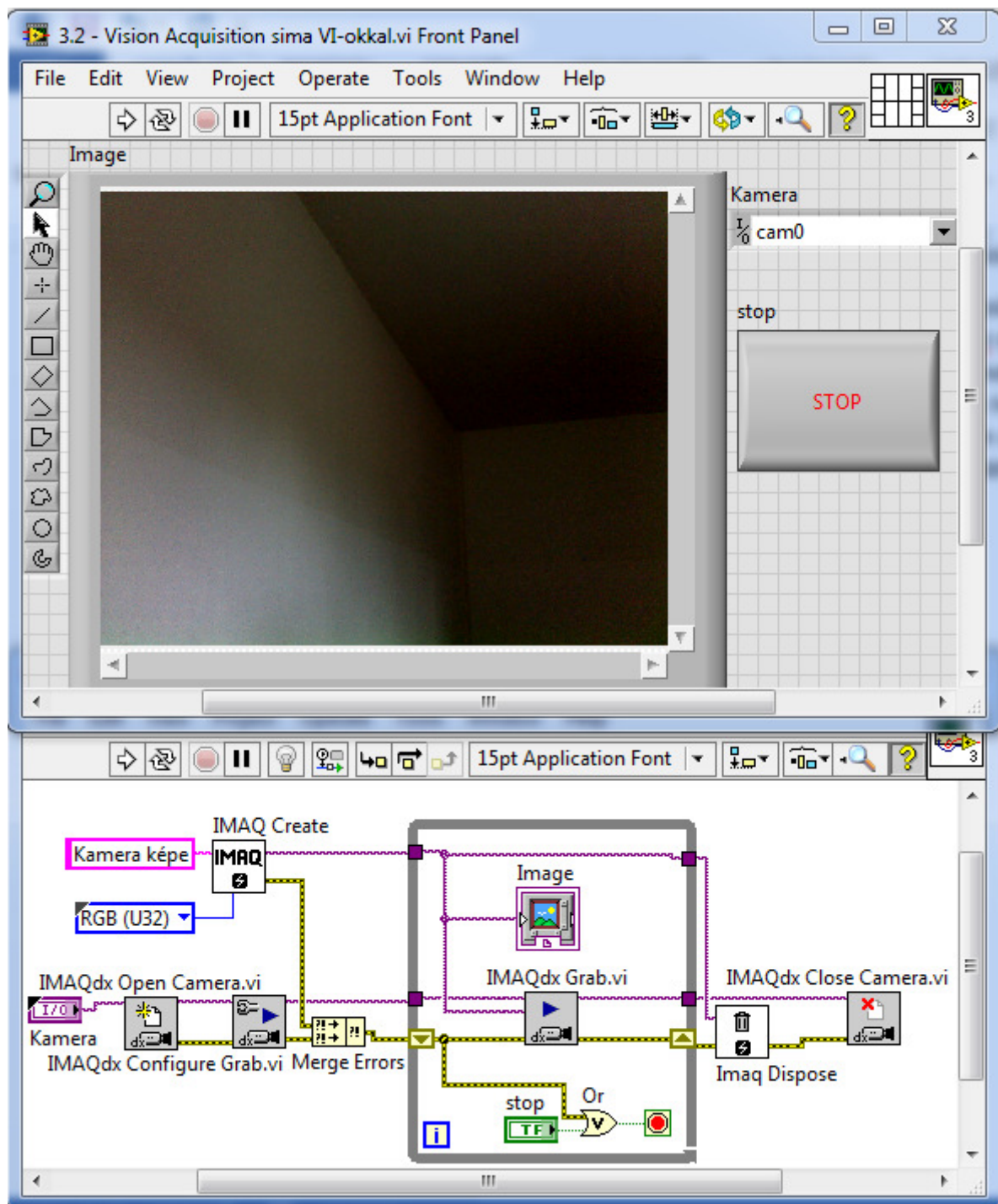
Természetesen a kamera képének beállítása végrehajtható klasszikus (nem expressz) VI-ok segítségével, bár ekkor több munkára (és blokkra) lesz szükségünk. Először is ki kell választani a Vision and Motion > NI-IMAQdx alatt található IMAQdx Open Camera.vi-t. Ez Session In bemenetén várja a használandó kamera sorszámát. Ezt a legegyszerűbben úgy tudjuk beállítani, ha rákattintunk jobb gombbal a bemenetre és Create > Control vagy Create > Constant opciót választunk. Első esetben megjelenik egy vezérlő a Front Panelen, ahol lenyíló menüből választhatjuk ki a kameránkat, míg második esetben a Block Diagramon jelenik meg ugyanez a lenyíló menü, ebben az esetben viszont még futtatás előtt ki kell választanunk a kamerát, így én inkább az első megoldást javaslom.

Ezután szükségünk lesz egy Image referenciára, hogy utána abban tároljuk majd a kamera képét. Ezért tegyünk le egy IMAQ Create VI-t a Vision and Motion > Vision Utilities > Image Managment alól. Bemenetként várja a kép nevét (Image Name), adjunk meg egy tetszőleges nevet. Fontos, hogy ha több Image referenciát hozunk létre a programunkban, akkor mindegyiknek egyedi neve legyen. Ezen kívül opcionális bemenetként megadhatjuk a kép típusát (Image Type), ezt azért lehet hasznos megadni, mert néhány VI (tipikusan a forma alapú mintázatot keresők) fekete-fehér képet várnak és így biztosan azt fogják kapni. Hasonlóan az előző VI-nál említett kameraválasztóhoz itt is javasolt a Control vagy Constant létrehozása a bemenethez. Én a Constant-ot javaslom, mivel ezt nem kell átállítani túl gyakran. Állítsuk is be RGB (U32)-re, hogy színes képet kapjunk.

Ezután szükségünk lesz egy while ciklusra, aminek leállításához hozzunk is létre egy stop gombot rögtön. A while cikluson belül szükségünk lesz az IMAQdx Grab.vi-ra (az Vision and Motion > NI-IMAQdx alól). Ez a vi fogja mindig a kamera legfrissebb képét az image változóba beletenni. A Session In bemenetére kössük rá az IMAQdx Configure Grab.vi által adott sessiont, az Image In-re pedig az IMAQ Create által adott New Image-t. Ezután a Front Panelre tegyünk le egy megjelenítőt a Vision menü alól, pl. az Image Display-t. Ebben fog megjelenni a kamera képe, ezért a Block Diagramon helyezzük ezt a while cikluson belülre és kössük bele az image változót.

A while ciklus után szükséges lesz lezárunk a sessiont, felszabadítani a memóriát. Ezt az Imaq Dispose.vi-jal és az IMAQdx Close Camera.vi-jal érhetjük el. Az első az image változót várja bemenetként, utóbbi a session-t. Mindkettő az előzőekkel azonos helyen érhető el.

Végezetül a hibakezelés miatt kössük össze az error be-és kimeneteket az ábrán látható módon. A while ciklus előtt a Merge Errors.vi segítségével fűzzük össze a két hibát, majd a while ciklusban Shift registerbe kötve használjuk, így minden cikluskör végén a következő cikluskör megkapja az előző esetleges hibáját. E hibakezelés nélkül ideális esetben fut a program, viszont hiba esetén, pl. ha a kamera nem elérhető, akkor a többi vi már nem fog lefutni, leáll a program. Épp ezért kell a stop gombhoz is egy or függvénnyel bekötni az Error Clustert.

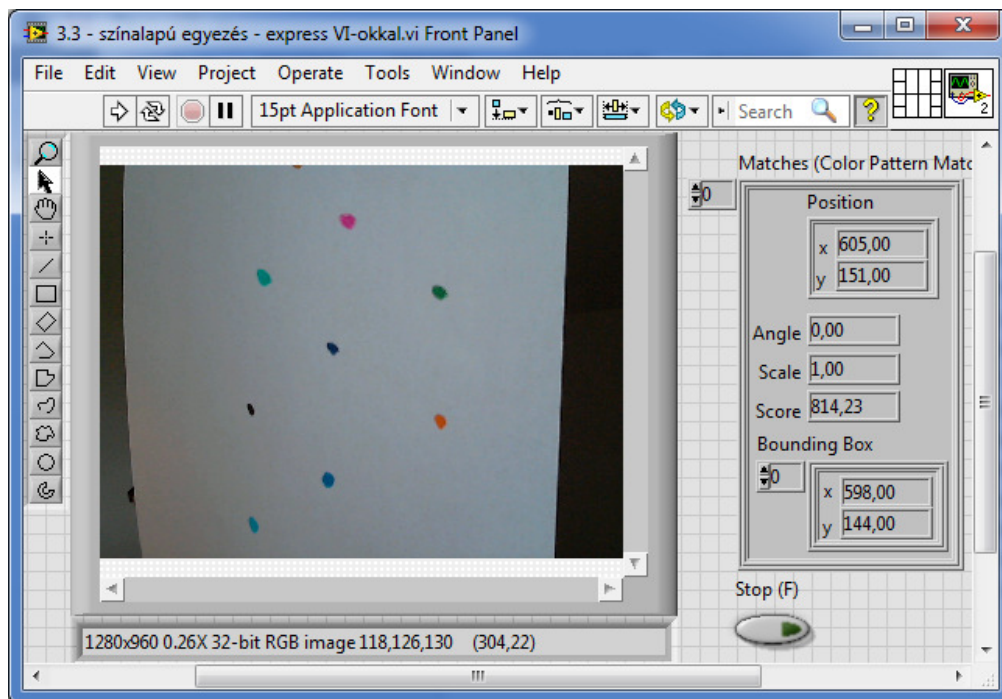


3.3.1. ábra - Vision Acquisition sima VI-okkal

3.4. Szín alapú egyezés keresés express VI-ok segítségével

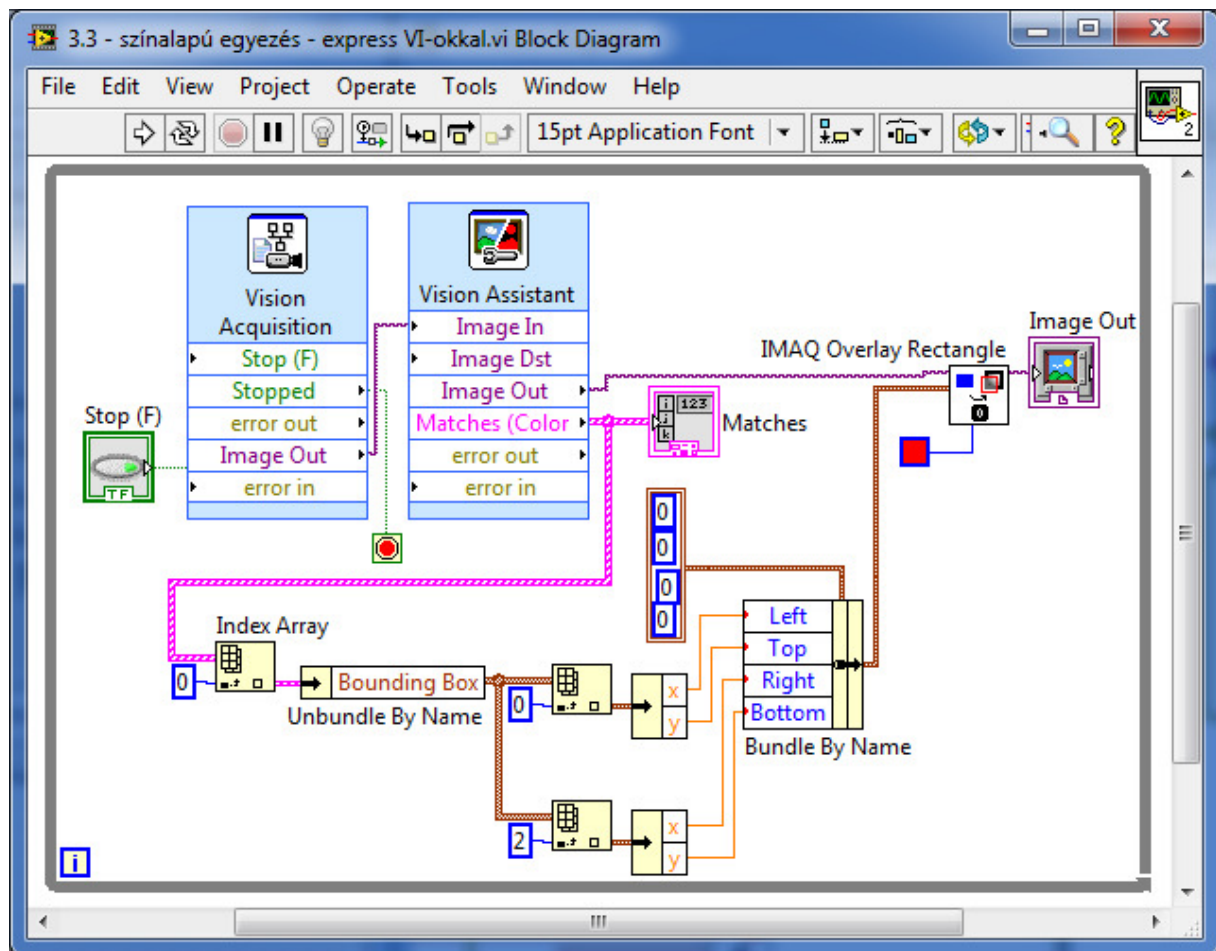
Kiindulásnak én az előzőleg lekészített *3.1 - Vision Acquisition Express VI-al.vi* programomat használtam. Először tegyük le a képfeldolgozást végző Vision Assistant express vi-t a Vision and Motion > Vision Express alól. Hasonlóan a már használt Vision Acquisition express vihoz itt is rögtön megnyílik egy varázsló, amit viszont azt mondja, hogy „There is no available image to process.”. Ezért kattintsunk a Finish-re és kössük rá a Vision Assistant Image In bemenetére

a Vision Acquisition Image Out-ját. Ezután indítsuk el a programot, és mutassuk meg a kamerának a mintázatot, majd állítsuk le a programot. Fontos, hogy a leállítás pillanatában a kamera képén rajta legyen a keresendő szín, mert az utolsó képet fogjuk felhasználni a beállítás során. Ezután kattintsunk kétszer a Vision Assistantra, ezzel újra megnyitva a varázslót, ahol most már megjelenik a kamera által látott utolsó kép. Bal lent a Processing Functions alatt válasszuk a második fület (Color) majd itt a Color Pattern Matchingot. Ezután megnyílik a Color Pattern Matching Setup. Az első fülön (Main) nevet adhatunk a lépésnek. Ennek akkor van nagyobb szerepe, ha egymást követően több képfeldolgozási feladatot hajtunk végre, amik közül az első csak például kijelöli a vizsgálat helyszínét és a későbbiek majd ott vizsgálódnak. Most hagyhatjuk alapbeállításon. Második fülön (Template) hozhatjuk létre a keresett mintázatot. Menjünk is rá a Create Template-re. Itt rajzoljuk körbe egy kis téglalappal a keresendő mintázatot, majd kattintsunk az OK gombra. Ezután el kell mentenünk a mintázatot képként. Ennek előnye, hogy így ugyanezt a mintát már nem kell még egyszer körberajzolni téglalappal, hanem betölthetjük fájlból is. Ezután visszakerülünk a Color Pattern Matching Setup ablakba. Itt a harmadik fülön (Settings) állíthatjuk be, hogy hány egyezést keressen (Number of Matches to Find), hogy hány ezrelékes egyezés legyen elegendő (Minimum Score), hogy ebben az egyezésben a szín mennyire játsszon szerepet (Color Score Weight). Ezen kívül beállítható a színérzékenység (Color Sensitivity) és a mintakeresési stratégia (Search Strategy). Végezetül beállíthatjuk azt is, hogy keresse-e elforgatva is a mintázatot, és ha igen, akkor +/- hány fokig. Az utóbbi beállításoknak abban az esetben van nagyobb szerepük, ha nem csupán egyetlen színt jelöltünk ki, mint mintát, hanem több színből álló mintázatot. Ezeket a beállításokat hagyhatjuk is így és kattintsunk az OK gombra. Ezzel visszatérünk a Vision Assistant alap ablakába. Itt menjünk a Select Controls gombra. A megjelenő ablakban választhatjuk ki, hogy az express vi-nak milyen be- és kimenetei legyenek. Bemenetként a korábban beállított beállítások szinte mindegyike kiválasztható, így futás közben is módosíthatjuk azokat. Kimenetként a Matches (egyezések) mindenképp szükséges, hiszen így kapjuk meg a képfeldolgozás eredményét. Emellett még esetleg a Number of Matches-re lehet szükségünk, ami a mintafelismerések számát adja meg. Most elegendő lesz nekünk a Matches. Menjünk a Finishre, és így sikeresen beállítottuk az express vi-unkat. Csináljunk is gyorsan egy indikátort a Matches kimenetre. Látható, hogy ez egy clusterekből álló tömb. Ha elindítjuk a programot, akkor találat esetén ki fogja írni a találat adatait: koordinátáit, az elfordítás szögét, a méretét, az egyezés mértékét ezrelékben, és a befoglaló négyzet adatait (esetemben a rózsaszín folt volt a mintázat):



3.4.1. ábra - színelapú egyezés - express VI-okkal – első verzió

Ez azonban még mindig nem az igazi. Sokkal látványosabb lenne, ha körberajzolná az egyezést. Ezért egészítsük a ki következővé a Block Diagramot:



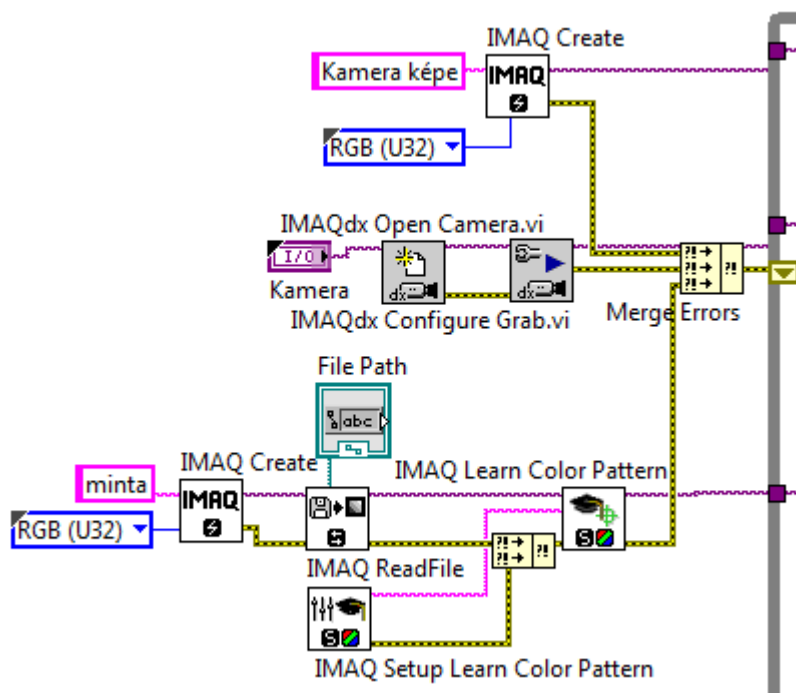
3.4.2. ábra - színelapú egyezés - express VI-okkal – végleges

A Vision Assistant által adott adatcsomag egy tömb, így először vegyük annak az első elemét. Így egy cluster-t kapunk. A cluster-ből nekünk a befoglaló téglalapra (Bounding Box) van szükségünk, így az Unbundle By Name parancs segítségével vegyük ezt az elemet. Ez ismét egy clusterekből álló tömb, aminek a 4 eleme a négyzet 4 sarkát takarja. Így nekünk az első és a harmadik kell. Ezután ismét egy Unbundle By Name paranccsal nyerhetjük ki a tényleges számokat. Ezeket viszont a Bundle By Name paranccsal össze kell fűznünk egy clusterré, mert a téglalaprajzoló IMAQ Overlay Rectangle clusterként várja a téglalap koordinátáit. A téglalap színéhez én kreáltam egy konstanst, amit így csak a Block Diagramon tudok állítani. Így tökéletesen működik, rárajzolja a kis piros négyzetet az egyezésre.

3.5. Szín alapú egyezéskeresés sima VI-ok segítségével

Természetesen az egyezéskeresést is meg lehet valósítani sima, nem express vi-ok segítségével. Ehhez kiindulásnak az előzőleg elkészített 3.2 - Vision Acquisition sima VI-okkal.vi-t veszem alapul. Először egészítsük ki a while ciklus előtti részt a színkeresés beállításával. Ehhez a Vision and Motion > Machine Vision > Searching and Matching alatt található IMAQ Setup

Learn Color Pattern és az ugyanitt lévő IMAQ Learn Color Pattern vi-okra lesz szükségünk. Az IMAQ Learn Color Pattern bemenetként várja a keresendő mintázatot képként. Én ezt egy külső fájlból töltöm be (ahova az express vi-os verzióban lementettem). A betöltéshez szükségünk lesz egy IMAQ Create által generált Image-re, majd ebbe tudjuk beletenni az IMAQ Read File (Vision and Motion > Vision Utilities>Files) segítségével. Az IMAQ Create beállításairól már beszéltem a 3.3-as fejezetben. Az IMAQ Read File bemenetként várja az IMAQ Create által létrehozott Image-t és a betöltendő fájl elérési útját (Path). Erre én egy Controllt hoztam létre, így a Front Panelen lehet tállózni a képet. Ezen kívül megadhatóak neki még opcionális bementként beállítások (File Options, Load Color Palette), amikkel finomhangolhatjuk a betöltést. Ezeket hagyhatjuk alapbeállításon és csak ha valami speciális igényünk van, vagy nem működik a program, akkor érdemes hozzájuk nyúlni.



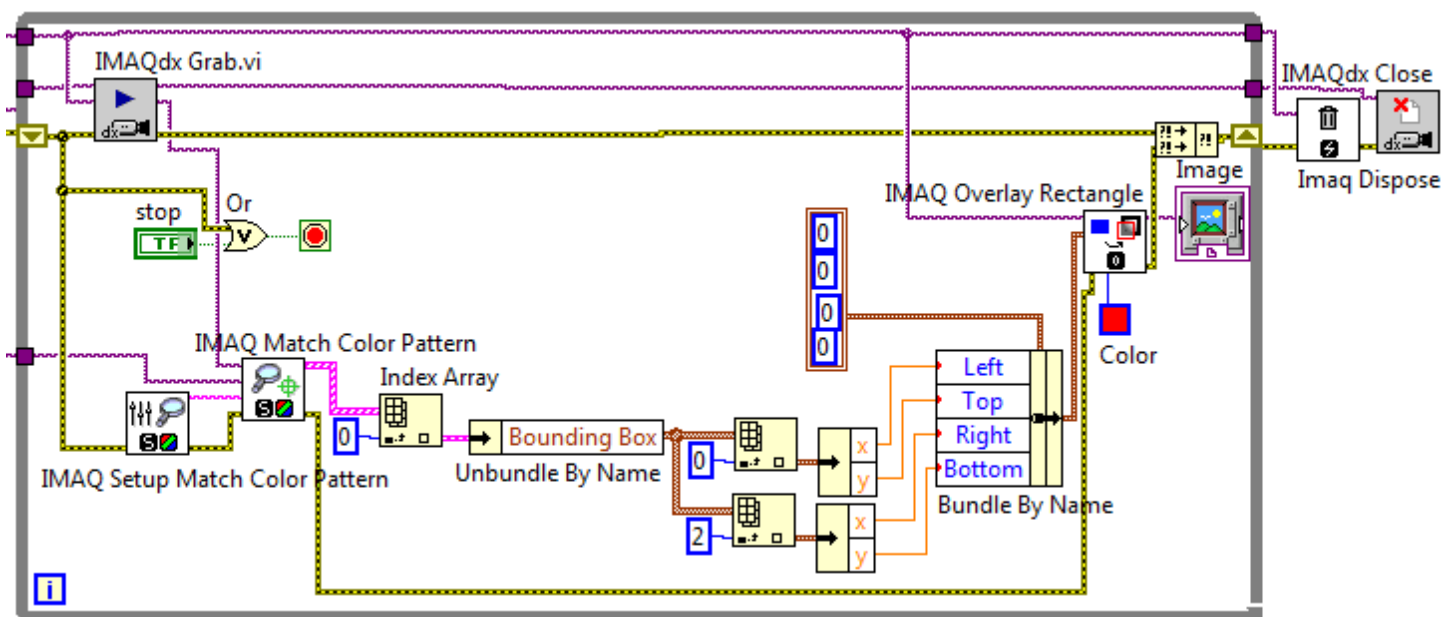
3.5.1. ábra - színapapú egyezés - sima VI-okkal - Block Diagram 1

Az IMAQ Read File által adott Image kimenetet kössük bele az IMAQ Learn Color Pattern Image bemenetére. Az IMAQ Setup Learn Color Pattern vi rengeteg opcionális bementén várja az egyezés tulajdonságait, hasonlóan azokhoz, amiket az Express vi esetén és beállíthattunk (pl. Lear Mode). Kimenetként ad egy Learn Color Pattern Setup Data adatot, amit az IMAQ Learn Color Pattern vár a hasonló elnevezésű bementére. A továbbiakban az IMAQ Learn Color Pattern által adott Template Image Out adatra lesz szükségünk, így ezt kössük be a while ciklusba.

Végezetül a hibakezelés miatt kössük össze az Error Out és Error In csatlakozókat, ahol szükséges Merge Error vi-jal összegezzük a hibákat. Ekkor a while ciklus előtti rész a jobb oldali ábrán látható módon fog kinézni.

A while cikluson belül a most definiált mintázat keresését a Vision and Motion > Machine Vision > Searching and Matching alatt található IMAQ Match Color Pattern végzi, amihez az

IMAQ Learn Color Pattern-hez hasonlóan létezik egy Setup vi is, IMAQ Setup Match Color Pattern. A Setup ismét rengeteg opcionális bemenetet kínál a beállításra, többek között itt állíthatjuk be, hogy elforgatva is keresse-e a mintát, vagy hogy milyen keresési stratégiával (Search Strategy) keressen. Az általa adott Match Color Pattern Setup Data-t várja ugyanilyen néven az IMAQ Match Color Pattern vi. Ezen kívül néhány beállítást is az IMAQ Match Color Pattern vi-nak kell megadni, például a találatok kívánt számát (Numer of Matches Requested) vagy, hogy hány ezrelékes egyezés legyen elegendő (Minimum Match Score). Az IMAQ Match Color Pattern ezen kívül bementként várja a keresendő mintázatot, a Template Image-t és a kamera képét. Kimenetként pedig egy ugyanolyan Clustert ad, mint a 3.4-ben bemutatott express vi. A clusterből a befoglaló téglalap kinyerését is onnan másoltam teljes egészében, így annak bemutatására nem térek ki újból. A while ciklus után a lezárás és memória felszabadítás a 3.2-ben ismertetett módon zajlik, így arról sem ejtek több szót. Végezetül így néz ki a Block Diagramm while ciklusa és a ciklus utáni rész:

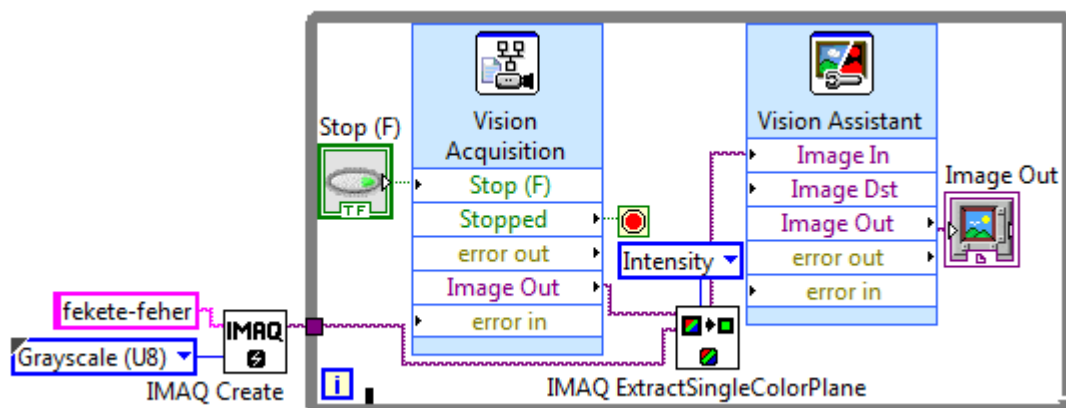


3.5.1. ábra - színelapú egyezés - sima VI-okkal - Block Diagram 2

3.6. Forma alapú egyezés keresés express VI-ok segítségével

Kiindulási alapként nyissuk meg a 3.3 - színelapú egyezés - express VI-okkal.vi-t és módosítsuk ezt. Először is a forma alapú egyezéshez nem színes, hanem fekete-fehér képet kell használnunk. Ezért a Vision Acquisition által adott színes képet a Vision and Motion > Vision Utilities > Color Utilities alatt található IMAQ ExtractSingleColorPlane vi segítségével tesszük monokrómmá. Ez a vi bementként várja a Vision Acquisition Image Out-ját mint Image Src (Source – forrás), valamint a Color Plane-t, ami esetünkben annak felel meg, hogy mi alapján

történjen a fekete-fehérré tétel. (Igazából ez az ExtractSingleColorPlane vi nem szabályos konvertálást hajt végre, csupán veszi az egyik color plane, pl a kép RGB kódolásából a zöldet, pirosat vagy a kékét, vagy a HSL kódolásból valamelyiket és csupán annak az egy rétegnek az információját viszi tovább. Viszont kiválaszthatjuk itt az Intensity opciót, ami összességében mégis az elvárt módon fogja fekete-fehérré konvertálni a képet.) A Color Plane bemenetre csináljunk egy konstanst vagy egy kontrollt és válasszuk ki az Intensity opciót. Ezután az ExtractSingleColorPlane vi Image Dst Out kimenetét kössük rá a Vision Assistant Image In-jére. Így azonban a memóriában ugyanazon a képe dolgozik a Vision Acquisition, az ExtractSingleColorPlane és a Vision Assistant valamint az Image Out megjelenítőben is ez látszik. Így azonban ha az ExtractSingleColorPlane vi kicsit késik előfordul, hogy egy pillanatra színes kép jelenik meg. Hogy ezt elkerüljük létre kell hoznunk az IMAQ Create-tel egy új fekete-fehér (Grayscale(U8)) képet és ezt megadni az ExtractSingleColorPlane mint Image Dst (destination – cél). Természetesen az IMAQ Create-t a while ciklus elé kell elhelyezni, hiszen csak egyszer szeretnénk, hogy lefusson.



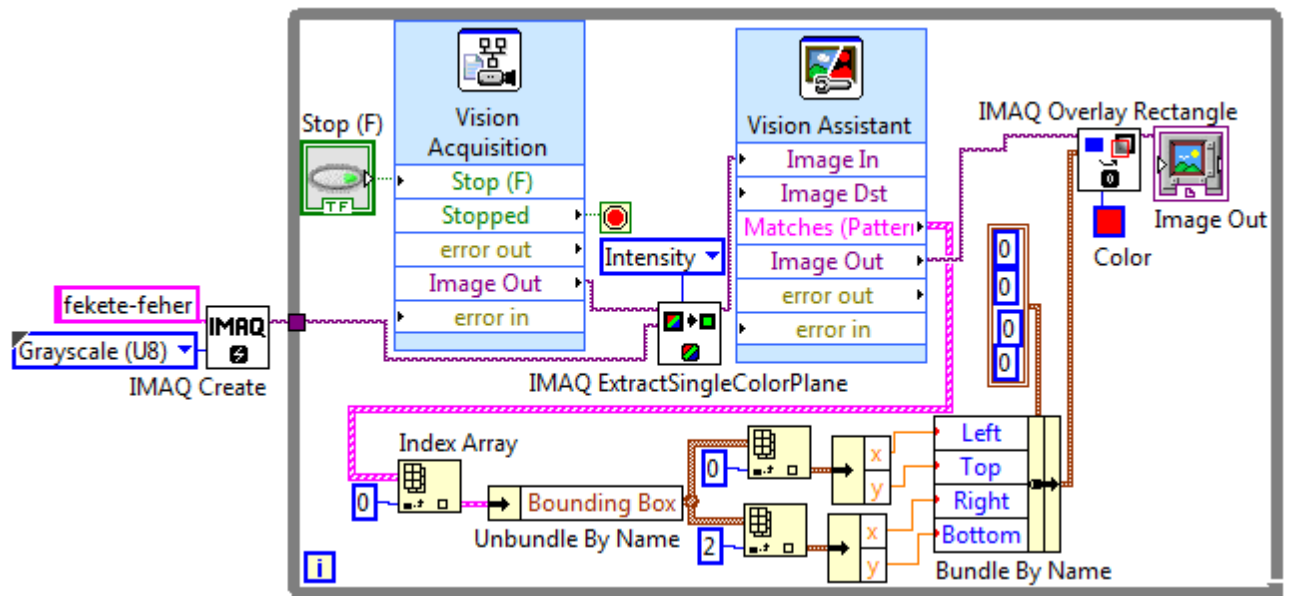
3.6.2. ábra - Forma alapú egyezéskeresés express VI-ok segítségével - Block Diagram 1

Ezután lépünk be a Vision Assistantba (dupla kattintás) és töröljük az alsó sorból a Color Pattern Matching-ot (jobb gomb > Delete), majd a Finish-el térünk vissza a Block Diagrammra. Erre azért van szükség, mert megint meg kell majd mutatnunk a keresendő mintát a kamerának. Viszont ha törlés nélkül indítanánk a programot, akkor a Vision Assistant hibával kilépne, mert fekete-fehér képen keresünk színeket. Így viszont a Matches clustert már nem adja tovább, így töröljük a téglalaprajzolás részt. Ekkor a feljebb látható módon fog kinézni a ciklusunk.

Indítsuk el ekkor a programot. Ha mindent jól csináltunk, akkor a képmezőben fekete-fehéreként fog megjelenni a kameránk képe. Mutassuk meg a kamerának a mintázatot majd stoppal állítsuk le a programot. Lépünk be ismét a Vision Assistant-ba. Bal alsó dobozban válasszuk az ötödik fület (Machine Vision) majd itt válasszuk a Pattern Matching opciót. Beállítása nagyon hasonlít 3.4-ben ismertetett módszerhez, így itt csak röviden áttekintem. New Template-tel csináljunk egy új mintázatot. A felugró ablakban keressük meg és rajzoljuk a keresendő forma köré egy téglalapot, majd menjünk a Next-re. A következő ablakban ecsettel kisatírozhatjuk azokat a részeket, amiket nem szeretnénk, hogy figyelembe vegyen a program az egyezéskereséskor. Rajzoljuk körbe a mintát, ha van benne lyuk, azt is „színezzük be”. Ha valamit véletlenül beszíneztünk, amit nem kellett volna, használjuk a radír eszközt és távolítsuk el a színezést. Ezen kívül a Ctrl+Z, Visszavonás is működik. Végezetül a jobb oldalon látható képhez hasonlót kapunk. Ekkor menjünk a Next-re. Előfordulhat, hogy túl kevés területet adtunk meg a programnak, vagyis vagy túl kicsi volt az első lépésben rajzolt téglalap, vagy túl nagy részét zártuk ki színezéssel. Erről hibaüzenetben értesít minket és nem enged továbblépni. Ekkor csökkentjük a beszínezett területek nagyságát vagy rajzoljunk nagyobb téglalapot. Ha sikerrel továbbléptünk, akkor jelölhetjük ki a koordinátarendszerünk középpontját. Ennek akkor van nagy szerepe, hogyha több lépésben végzünk képfeldolgozást, és első lépésben csak a vizsgálandó területet határozzuk meg, rögzítünk hozzá egy koordináta rendszert, és a későbbi lépések már ehhez a koordinátarendszerhez képest dolgoznak csak. Esetünkben hagyhatjuk ezt alapbeállításon és kattintsunk a Finish-re. Ekkor ismét el kell mentenünk valahova az elkészített mintázatot, ezt tegyük is meg. A Settings oldalon még opcionális jelöljük be, hogy elforgatott mintázattal is keressen (Search for Rotated Patterns). Ezzel a beállítással azonban vigyázzunk, sokkal erőforrásigényesebb lesz tőle a programunk, így csak indokolt esetben használjuk! Ezután kattintsunk az OK gombra. Ezzel visszakerültünk a Vision Assistant főmenüjében. Itt menjünk a Select Controls gombra. Ismét a 3.4-ben megismert ablakhoz hasonlót látunk. Pipáljuk be a Matches-t az indikátorok között és menjünk a Finish-re. Most nyissuk meg a 3.3 - színalapú egyezés - express VI-okkal.vi-t és az előzőleg törölt Matches-ből téglalaprajzoló részt másoljuk vissza, majd kössük be a következő ábrán látható módon:



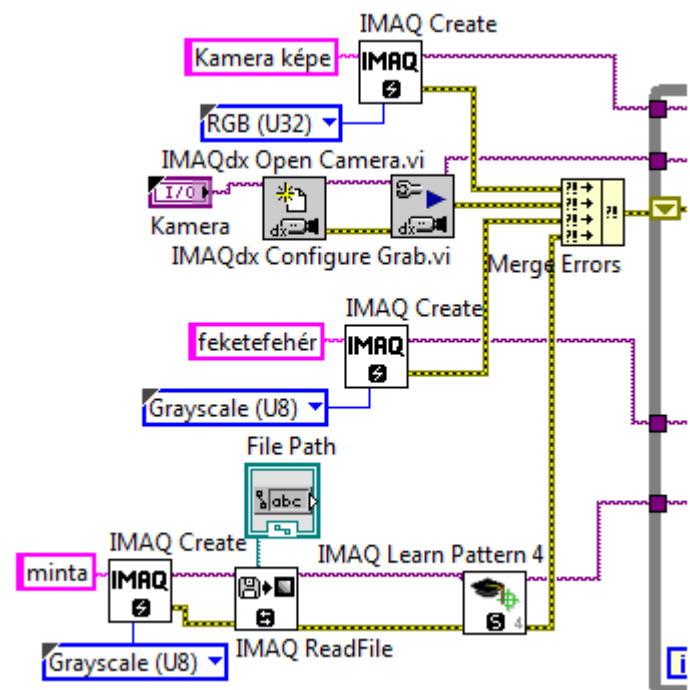
3.6.3. ábra - - Forma alapú egyezéskeresés express VI-ok segítségével - Mintázat



3.6.4. ábra - Forma alapú egyezéskeresés express VI-ok segítségével - Block Diagram 2

3.7. Forma alapú egyezéskeresés sima VI-ok segítségével

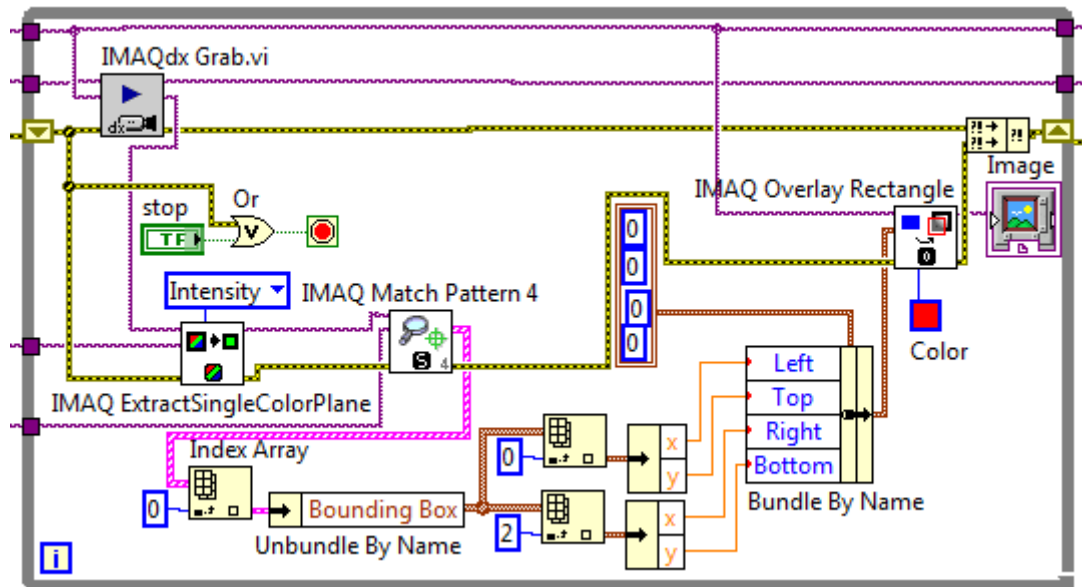
Kiindulásnak használjuk fel a 3.4 - színelapú egyezés - sima VI-okkal.vi-t. A while ciklus előtti részt módosítjuk először. Most a 3.6-ban megismert, forma alapú egyezéskereséshez hasonlóan itt is szükségünk lesz egy fekete-fehér képre. Ezért az IMAQ Create segítségével csináljunk egyet, Image Type-nak Grayscale (U8)-at adjunk meg. Állítsuk át a minta Image típusát is Grayscale (U8)-ra. Ezután az IMAQ Learn Color Pattern és az IMAQ Setup Learn Color Pattern helyett csupán egyetlen vi-ra lesz szükségünk, az IMAQ Learn Pattern 4-re (Vision and Motion > Machine Vision > Searching and Matching). Ismét opcionális bemenetként rengeteg beállítást megadhatunk, de ezek bemutatásától most eltekintek. Most körülbelül a jobb oldalt látható módon néz ki a Block Diagram while ciklus előtti része.



3.7.1. ábra - Forma alapú egyezéskeresés sima VI-ok segítségével - Block Diagram 1

A while cikluson belül egyrészt szükségünk lesz arra, hogy a kamera képét fekete-fehérré tegyük és ezt a képet eltároljuk a feketefehér néven létrehozott Image változóban. Ehhez a 3.6-

ban megismert IMAQ ExtractSingleColorPlane vi-t használjuk. Ennek leírására már a 3.6-ban kitértem, így itt nem ismételtem meg. Ezen kívül természetesen itt is le kell cserélnünk az IMAQ Match Color Pattern és az IMAQ Setup Match Color Pattern vi-okat. Ezek helyett az IMAQ Match Pattern 4-t fogjuk használni (Vision and Motion > Machine Vision > Searching and Matching). Szerencsére ez ugyanolyan formában adja vissza az egyezéseket, mint a Match Color Pattern, így egyszerűen csak kössük be a Matches (pixels) kimenetet és már működni is fog a téglalaprajzolás. Ekkor így fog kinézni a while ciklus:

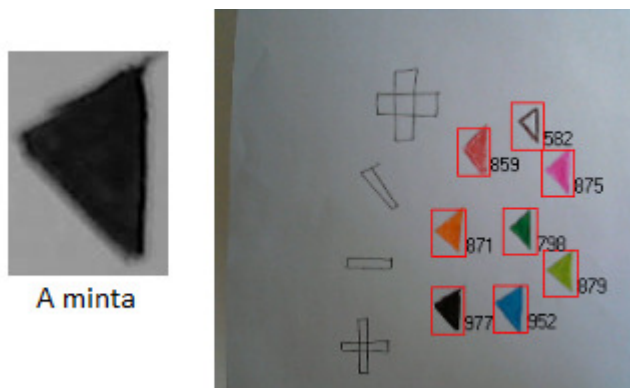


3.7.2. ábra - Forma alapú egyezéskeresés sima VI-ok segítségével - Block Diagram 2

Figyeljük meg, hogy ha futtatjuk a programot, akkor a színes ábra jelenik meg és arra is rajzolja az egyezést jelentő téglalapot. Ennek oka, hogy az IMAQ Overlay Rectangle-nek bemenetként az eredeti, színes képet adjuk meg, nem pedig az általunk kreált fekete-fehéret. Mivel átméretezés nem történt, így a pixel koordináták ugyanott vannak mindkettőn, így jó helyre kerül a színes képen is a fekete-fehéren megtalált egyezés.

Kiegészítés

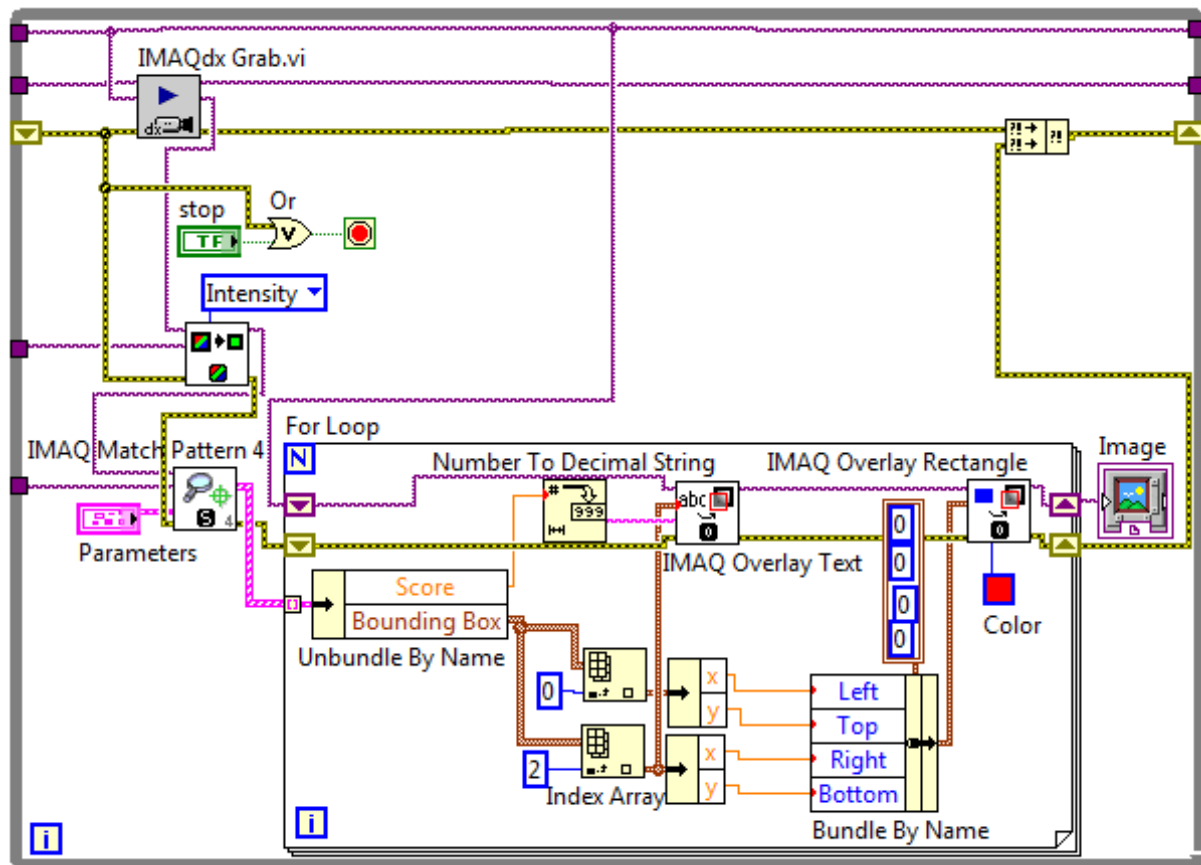
Egészítsük ki az előző vi-unkat úgy, hogy ne csak egy, hanem az összes egyezésre rárajzolja a



3.7.3. ábra - Forma alapú egyezéskeresés sima VI-ok segítségével – Kiegészítés I

téglalapot, valamint mindegyik találat mellé írja oda azt is, hogy hány ezrelékes az egyezés! Az elvárt eredmény a bal oldalt látható. Ehhez először is csináljunk egy Control-t az IMAQ Match Pattern 4-nak a Parameters bemenetére. Itt adhatjuk majd meg, hogy hány egyezést keressen (Number of Matches Requested) és hogy hány ezrelékes egyezés legyen elég (Minimum Match Score). Ezután a téglalaprajzoló részt

kell egy kicsit átalakítani. Az egyezéseket (Matches) egy tömbként kapjuk meg, aminek eddig mindig az első elemét vettük. Kössük rá ehelyett ezt a tömböt egy for ciklusra, ami így a tömb elemein, vagyis az egyezéseken fog végigmenni. A for ciklusnak ezen kívül kössük még be a kamera képét shift regiszterként, így minden körben az előző kör által már módosított képre fog még egy téglalapot rajzolni illetve írni. A for cikluson belül el kell helyezni az eddigi téglalaprajzoló blokkot, valamint gondoskodni kell a találati arány kijelzéséről is. Ehhez először is az Unbundle by Name vi-t nagyobbítsuk meg és válasszuk ki a Score változót. Ez a Score azonban egy double, így stringgé kell konvertálni a Number to Decimal String vi segítségével. Ezt követően történik a szöveg tényleges képre írása az IMAQ Overlay Text segítségével. Itt bemenetként meg kell adnunk a képet, amire írunk, felírandó szöveget (String) és az írás helyét pixelekben (Origin). Ehhez használjuk föl a befoglaló téglalap 3. sarkának koordinátáit, amit már a téglalaprajzolásnál megkaptunk. Ezen kívül természetesen opcionális beállításként megadhatnánk a szöveg betűtípusát, méretét, színét, de most jó lesz nekünk fekete szöveg alapértelmezett beállításokkal. Végül a hibakezelés miatt kössük össze az Error in-Error Out csatlakozókat és ennek is csináljunk egy Shift regisztert. Végezetül így fog kinézni az eredeti while ciklusunk:



3.7.4. ábra - Forma alapú egyezéskeresés sima VI-ok segítségével – Kiegészítés 2

4. Magasabb szintű képfeldolgozási technikák

4.1. Kétdimenziós vonalkód olvasás express VI-ok segítségével

Az eddig használt Vision Assistant express vi lehetőséget nyújt arra, hogy viszonylag egyszerűen valósítsunk meg a kétdimenziós vonalkódolvasást. Ehhez kiindulásul a 3.5 -



4.1.1. ábra - Kétdimenziós vonalkód olvasás express VI-ok segítségével

formaalapú egyezés - express VI-okkal.vi-t használom föl. Azért épp ezt, mert a vonalkódolvasás is fekete-fehér képen működik. Lépünk be a Vision Assistantba, és töröljük ki a már ott lévő Pattern Matching-ot. Ezután a bal alsó dobozban menjünk az utolsó fülre (Processing Functions: Identification) és válasszuk a 2D Barcode Reader opciót. A megnyíló ablakban választhatjuk ki, hogy mi legyen a művelet neve, illetve, hogy milyen 2D vonalkódot keressen (Data Matrix, PDF417 vagy QR Code). Én az QR Code-ot választom most a példámhoz.

A következő két fülön (Settings és Grading) különböző finomhangolási beállításokat állíthatunk, ezeket én most alapbeállításon hagyom. Menjünk az Ok gombra majd a Vision Assistant főablakában a Select Controls-ra. Megint megadhatunk neki futás közben néhány beállítást, de amit fontos kiemelni, hogy a vonalkód típusát itt már nem módosíthatjuk. Kimenetként feltétlen pipáljuk be a Barcode Report-ot, hogy megkapjuk a leolvasott vonalkód dekódolt tartalmát. Ezután menjünk a Finish-re. Annyi probléma merült föl, hogy szemben a 3.5-forma alapú egyezéssel, itt nem egy clusterekből álló tömböt kapunk, hanem csak egy darab clustert. Ezért az első Index Array felesleges. Töröljük és kössük össze az eredetileg bele és belőle futó vezetékeket. Most ha futtatjuk a vi-t és mutatunk neki egy QR kódot, akkor be fogja keretezni pirossal. Ha közelítjük-távolítjuk a kamerától, akkor szépen látszik, hogy követi a kód méretét a téglalap. Ahhoz, hogy a kód tényleges tartalmához is hozzáférjünk, nagyobbítsuk meg az Unbundle by Name-t és válasszuk ki a Data opciót, majd kreáljunk egy indikátort hozzá. Az eredményt futás közben feljebb láthatjuk, a program pontos kinézetét pedig itt:

5. Mélységi kép

EBBEN A FEJEZETBEN BEMUTATOTT PROGRAMOK SORÁN ERŐSEN TÁMASZKODTAM EGY MAGÁT KLEMEN NICKNÉVVEL ILLETŐ EMBER MUNKÁJÁRA (DECIBEL.NI.COM/CONTENT/BLOGS/KL3M3N). AZ ÁLTALA ÍRT PROGRAMOKAT VETTEM ALAPUL SAJÁT PROGRAMOM MEGÍRÁSAKOR, ÉS NÉHÁNY SUBVI-ÁT IS FELHASZNÁLTAM. AZ ÁLTALA ÍRT PROGRAM MINDENKI SZÁMÁRA ELÉRHETŐ A FORUMS.NI.COM/T5/MACHINE-VISION/STEREO-VISION-AND-PROJECTED-LIGHT/M-P/2481294#M39150 OLDALON. EMELLETT MUNKÁM SORÁN TANULMÁNYOZTAM A LABVIEW BEÉPÍTETT PÉLDAPROGRAMJAIT IS, AMELYEK (A KIEGÉSZÍTŐK TELEPÍTÉSE UTÁN) C:\PROGRAM FILES (X86)\NATIONAL INSTRUMENTS\LABVIEW 2013\EXAMPLES\VISION\STEREO VISION\MAPPÁBAN ÉRHEŐK EL. INNEN IS FELHASZNÁLTAM PROGRAMRÉSZLETEKET (PL. A MÉRÉS KÖZBENI PONTOS MÉLYSÉG KIJELZÉSÉT).

5.1. Bevezetés

Ahogy az emberi agy képes a két szem kicsit eltérő képből összerakni a mélységi képet, ugyanezt meg lehet valósítani két, horizontális eltolt kamerával és egy számítógéppel. Fontos itt azonban kiemelni, hogy az emberi látásnál nem csupán ezt az információt használja az agy, hanem emellett fontos szerepet játszik az is, hogy mely tárgyak vannak takarásban (amelyiket takarja egy másik az értelemszerűen messzebb van), hogy egy ismert tárgyat mekkorának látunk (minél kisebb, annál messzebb van), az árnyékok elhelyezkedése, illetve a textúragradiens (ahogy a megfigyelő távolodik, úgy csökken a felszín részletezettsége) is. Ezekkel a számítógépünk (az általam megvalósított programban legalábbis) nem tud számolni, csupán a két eltérő képből próbál a mélységre következtetni.

Az algoritmus röviden a következő lépésekből áll:

- kalibráció: az eltérő képek közötti különbségek feltérképezése. Egy ismert fekete-fehér rácsot mutatunk a kamerának, ebből generálja le először az egyes kamerák, majd a kamerarendszer tulajdonságait
- tényleges mérés: a kalibrációnál elmentett beállítások alapján a két kamera képéből mélységi képet illeszt össze a program

Fontos, hogy akár csak a szemnél ez a módszer is csak bizonyos határokon belül alkalmazható. Túl közel túlságosan el fog térni a két kép és nehezen fogja megtalálni az összeillő pontokat, míg túl távoli tárgyak esetén a két kép szinte teljesen meg fog egyezni, így szintén nem fogja tudni kitalálni a távolságot. Emellett még egy korlátot szab, hogy a kamerák sem tökéletesek, valamint elhelyezkedésük sem.



5.1.1. ábra - Sztereo rendszer elrendezése

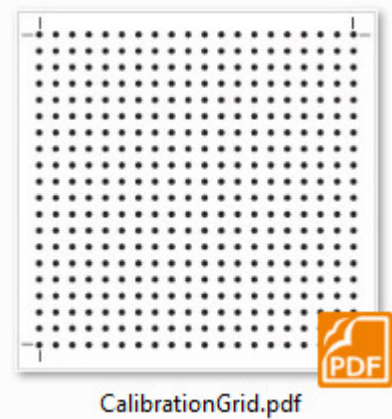
Nem mondhatjuk, hogy teljesen pontosan ugyanarra néznek, sem azt, hogy csak horizontálisan vannak eltolva. Az általam használt kameraelrendezés jobb oldalt látható. Kettő, egyforma Logitech C310-es webkamerát használtam, melyek közvetlen egymás mellett vannak rögzítve, így az objektívek távolsága 70 mm.

5.2. A sztereo rendszer kalibrálása

A sztereo rendszer működésének első lépéseként kalibrálnunk kell a kamerákat. Így tudjuk meg a rendszerünk paramétereit, amelyek egyrészt az egyes kamerák tulajdonságai, másrészt a kamerarendszer egészének tulajdonsága. A feladat bonyolultsága miatt ebben a fejezetben nem mutatok be minden egyes vi-t, csupán nagyvonalakban ismertetem, hogy programom mely része mit csinál.

Használat közben és a Front Panel

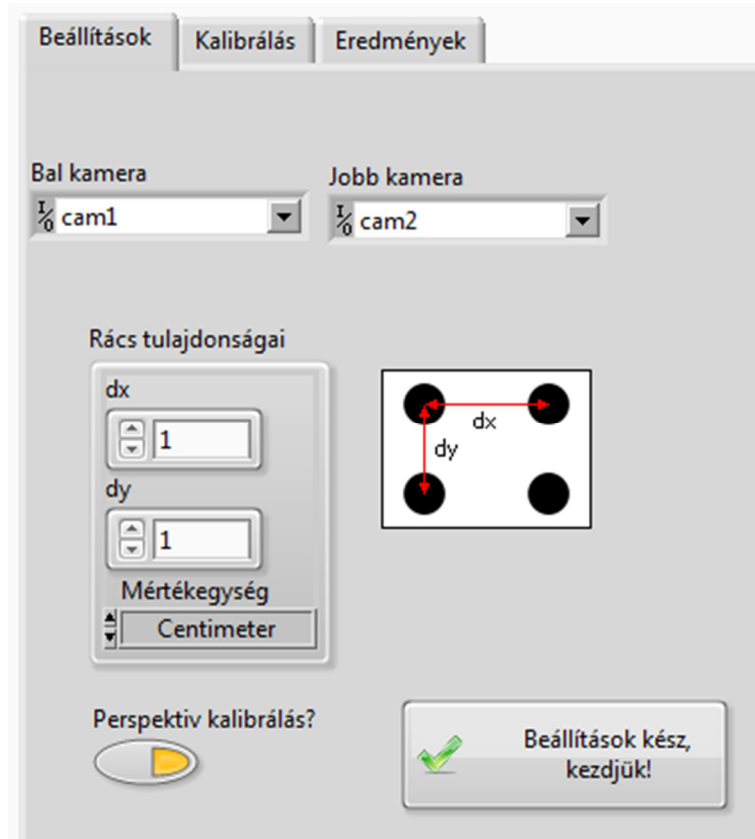
A kalibráláshoz a National Instrument által ajánlott kalibrációs rácsot (Calibration Grid) használok, ami a kiegészítők telepítése után elérhető a C:\Program Files (x86)\National Instruments\Vision\Documentation mappában CalibrationGrid.pdf néven. Voltaképp egy A4-es lap, amin 20x20 fekete pötty van. Ezt kell kinyomtatnunk és a kalibrálás során megmutatni a kameráknak. Ha 100%-os méretben nyomtatjuk ki, akkor a pontok között pont 1 cm lesz mindkét irányban (ez a kalibrálás során fontos adat lesz).



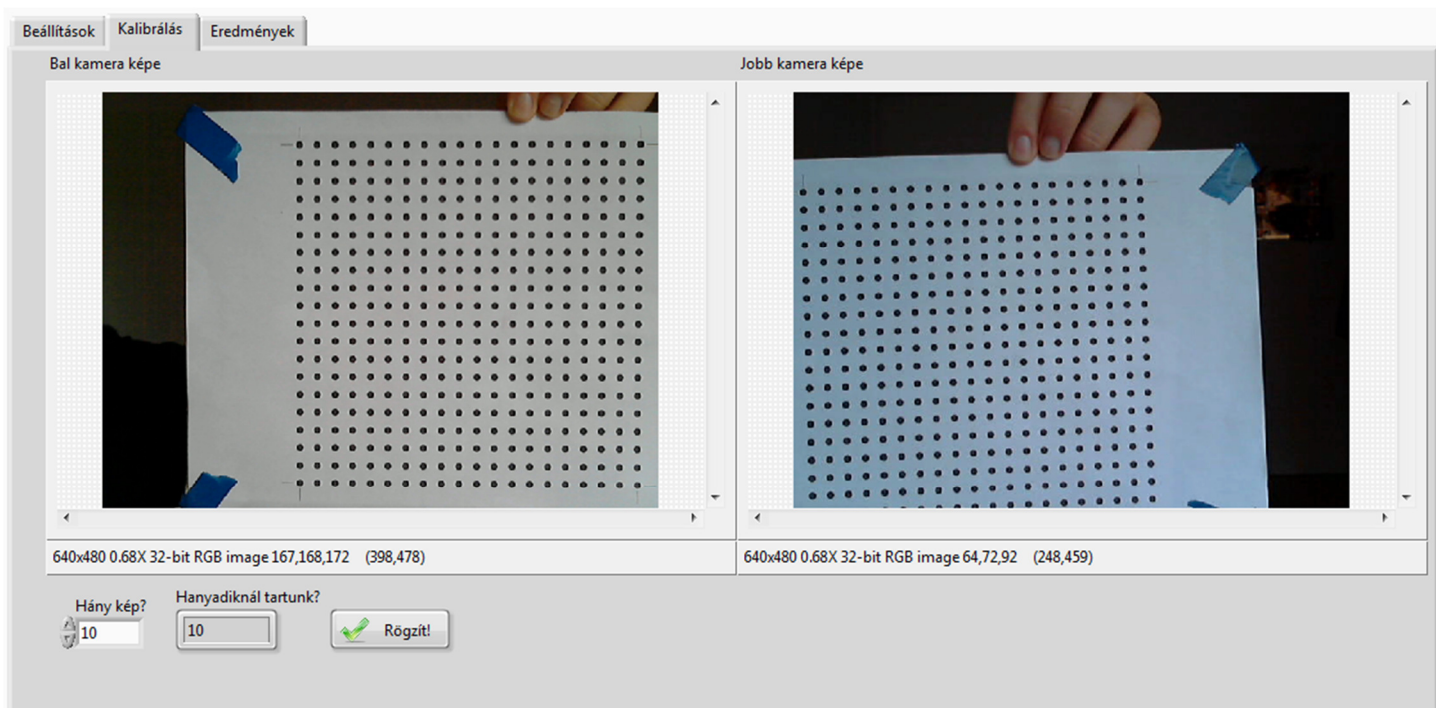
5.2.1. ábra - Kalibrálási rács

A kalibrálás elején ki kell választanunk a két kamerát. Itt fontos megjegyezni, hogy a kamerák szemszögéből érvényes a bal és a jobb, tehát ha szemben állunk a kamerákkal akkor a bal kezünkönél lesz a jobb kamera és a jobb kezünk felé fog esni a bal kamera. Ezután meg kell adnunk, hogy a kinyomtatott rácson x és y irányban mekkora távolságra vannak a pontok egymástól. A mértékegység lenyíló menüben tetszőleges metrikus vagy angol-szász mértékegységet választhatunk. Ezután eldönthetjük, hogy perspektív kalibrálást szeretnénk-e. Ekkor egy plusz vi-t is meghív a képekre, ami segít kijavítani az abból eredő hibákat, hogy a kamera nem merőleges pontosan a vizsgálandó terület síkjára.

Ha ezzel elkészültünk, mehetünk a Beállítások kész, kezdjük gombra. Ekkor a program automatikusan átugrik a Kalibrálás fülre, és bekapcsolja a kamerákat.



5.2.2. ábra - A sztereo rendszer kalibrálása - Beállítások

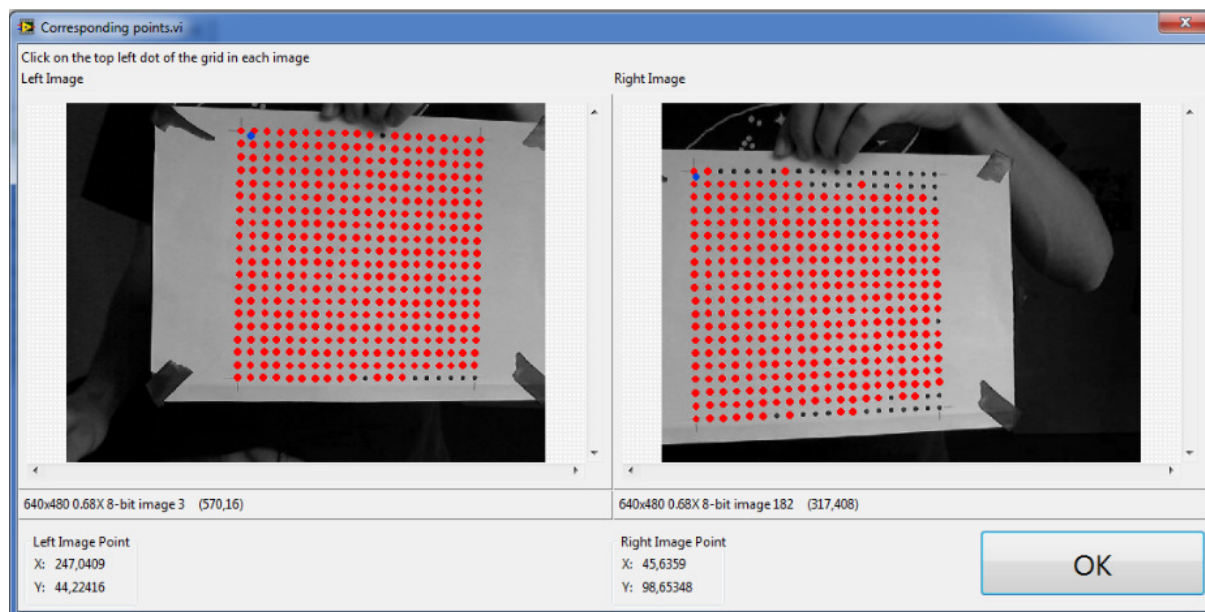


5.2.3. ábra - A sztereo rendszer kalibrálása - Kalibrálás

Rövid várakozás után megjelenik mindkét kameránk képe a képernyőn. Ekkor kell felmutatnunk az előzőleg kinyomtatott rácsot a kameráknak. Itt beállíthatjuk azt is, hogy hány darab képpel végezze a kalibrálást. Nekem 10 volt az ideális mennyiség. A kalibrálás úgy zajlik, hogy felmutatjuk a rácsot, és amikor úgy gondoljuk, hogy pont jól látszik mindkét kamerában, akkor megnyomjuk a Rögzít! gombot. Ennek hatására a program veszi az aktuális képet, fekete-fehérré tesz, megkeresi a rácsot (ezt jelzi is a felhasználónak, piros pontokat rajzol a felismert rácspontra) majd ennek segítségével elvégzi a kalibrálás első lépését és lementi a képet egy változóba.

A rács felmutatásánál fontos figyelembe venni, hogy az első kép a sztereo kalibráláshoz kell, tehát annál az a fontos, hogy mindkét rács teljes egészében látszódjon, vagy legalább a bal felső sarok mindenképp. A későbbi képek az egyes kamerákat állítják be, így ott az a célravezető, hogyha egyszerre csak egy kamerára koncentrálva lefedjük a ráccsal a látómezőjének először bal majd jobb felét és utána megismételjük ezt a másik kamerával is.

Ha végeztünk az előre megadott számú képpel, akkor a program feldob egy ablakot, ahol az elsőnek mintavételezett képen kell a bal felső pontot megjelölnünk (az ezt megvalósító Corresponding points.vi subVI Klemen munkája).



5.2.4. ábra - A sztereo rendszer kalibrálása - Corresponding points

Miután ezzel is megvoltunk, a program megkérdezi, hogy elmentse-e és ha igen, akkor hova a sztereo kalibrálási beállításokat. Majd ha ezzel megvolt, leáll és minket az Eredmények fülön hagy. Itt látjuk egyrészt a két kamera kiolvasott tulajdonságait (fókusz távolság, optikai középpont), a sztereo kalibrálás eredményét, és hogy a fájlba írás hiba nélkül lezajlott-e.

Ezzel meg is van a kalibrálási fájlunk, amit majd az 5.2 - stereo vision.vi-ban fogunk felhasználni.

A program működése, a Block Diagram

A Block Diagrammon egy hatalmas Flat Sequence Structure alkotja a program vázát. Ez felelős azért, hogy a program adott részei szépen egymást követően menjenek végbe. Első „filmkockában” a program várja, hogy a beállításokkal végezzen a felhasználó. A másodikban létrehozuk kalibrálás által használt képeket, és elindítjuk a kamerát. Itt felhasználtam egy speciális tulajdonságot (Property), amit a Class Browserből lehet előhívni. A Camera sessionben az IMAQdx class VideoMode tulajdonságát állítottam 17-re, ezzel csökkentve az amúgy HD minőségű kép méretét 640x480-ra. Erre azért volt szükség, hogy a kalibrálás gyorsabban végbemenjen, a rácspontok pedig ilyen felbontásnál is simán felismerhetők. Ezután kezdődik a tényleges mintavételezés, egy while ciklus pörög addig, amíg a felhasználó meg nem nyomja a Rögzít! gombot. Ezután az adott képet átalakítja a program fekete-fehérré és

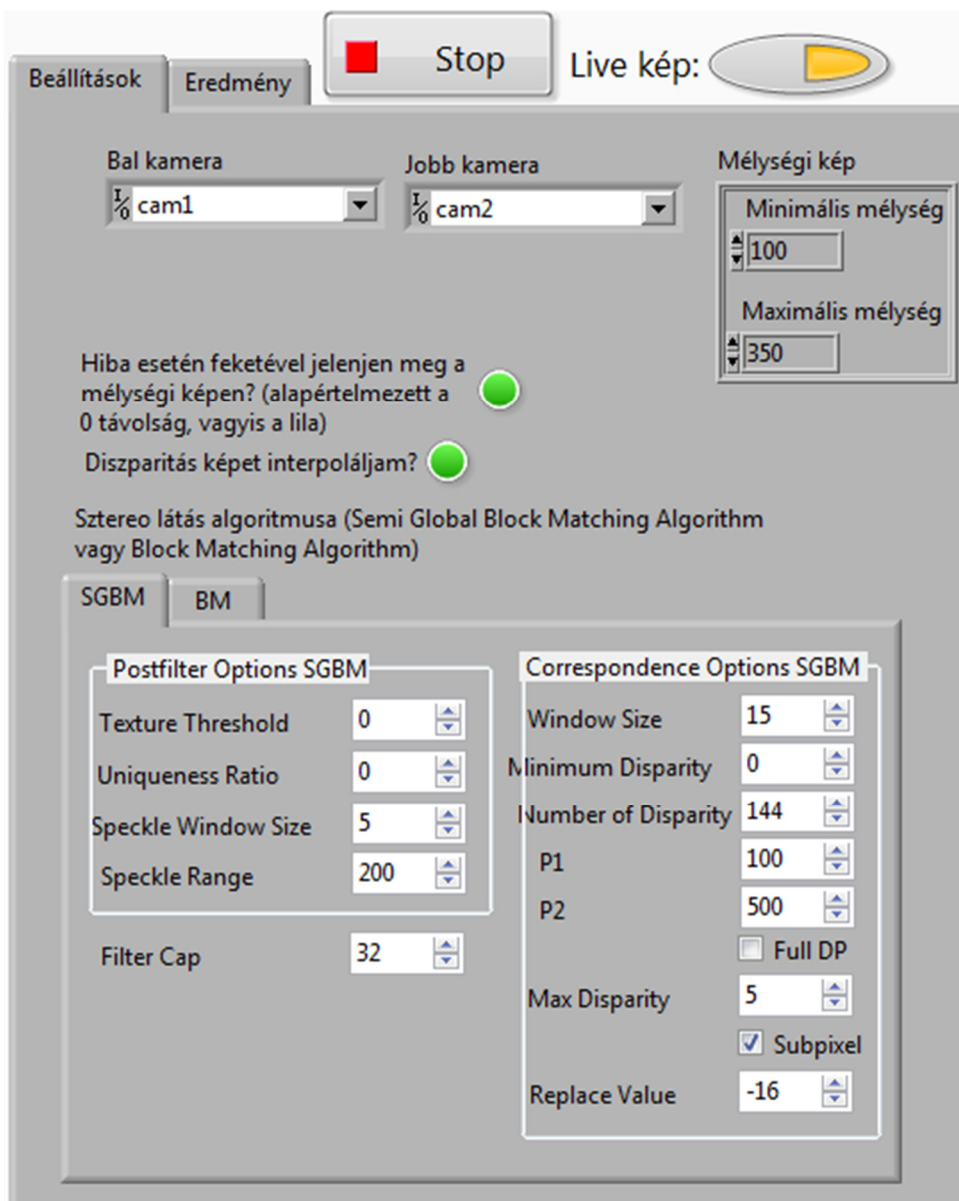
eltárolja egy most létrehozott változóba. Majd ráilleszti a rácsot (szintén egy Klemenés subVI segít ebben), ezután az IMAQ Learn Camera Model vi elvégzi az adatok tényleges kiolvasását, majd visszatér a mintavételezés elejére. Az eddig kalibrációs eredményeket kameránként külön képbe, a fekete-fehér Left Camera Calibration illetve Right Camera Calibration-ba tárolja. Későbbi futásoknál is ezt a képet módosítja. Miután megadottszámúszor végrehajtotta a mintavételezést tovább lép, a folyamatosan új változókba létrehozott képeket tömbként adva tovább. Ha a perspektív kalibrálás be van kapcsolva, akkor a program szétbontja ezt a tömböt és egyesével mindegyik elemére meghívja az IMAQ Learn Perspective Calibration vi-t, ami szintén a Left/Right Camera Calibration-t módosítja. Ezután már csak erre a két Calibration képre lesz szükségünk, ezért a többit törli a program. Ezután a Corresponding Points subvi feldobja az ablakot a felhasználónak, hogy jelölje ki mindkét képen a bal felső pontot. Utána ennek az információnak segítségével az IMAQ Learn Binocular Stereo Calibration.vi elvégzi a kalibrálását a kamerarendszernek. Még annyi van hátra, hogy a program rákérdezzen, hogy kívánjuk-e menteni, és ha igen hova, valamint, hogy ide el is mentse a kalibrálást az IMAQ Write Binocular Stereo File.

5.3. A sztereo rendszer használata

Használat közben és a Front Panel

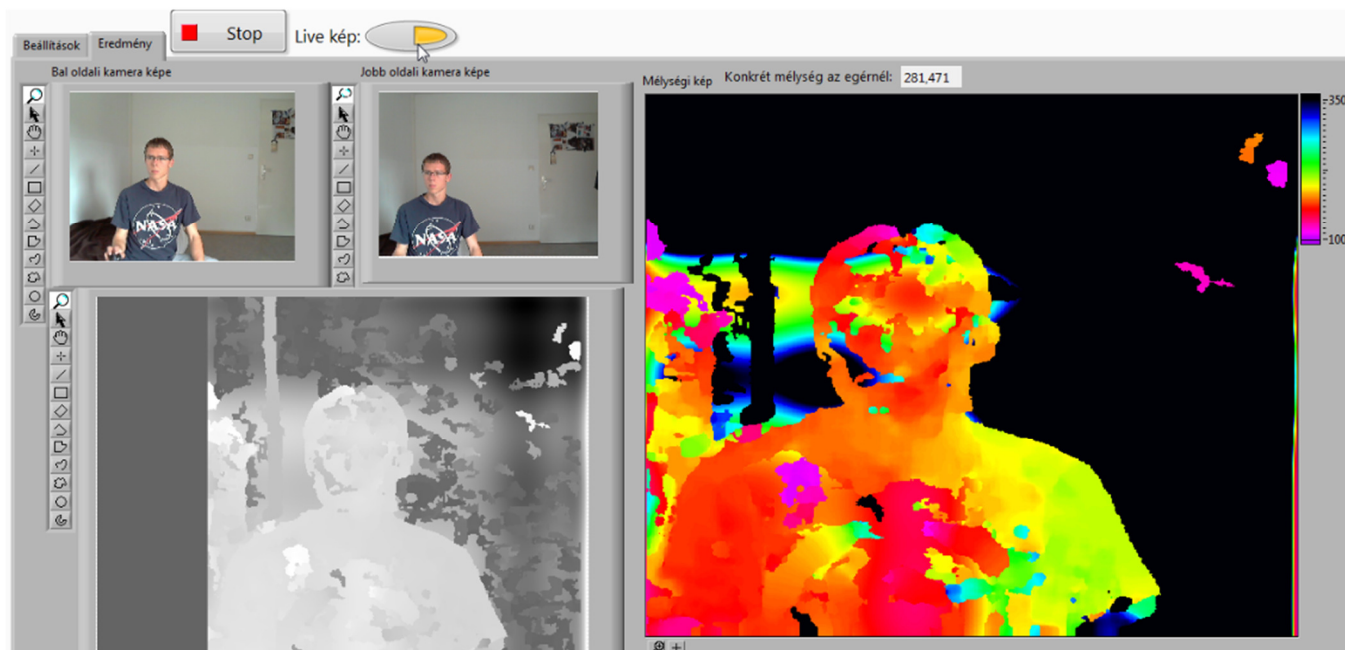
A programot praktikus még indulás előtt beállítani. A legfontosabb beállítások a két kamera kiválasztása, illetve, hogy a mélységi képen mely határok között keressen és jelenítsen meg dolgokat. Emellett beállíthatjuk, hogy amely pixelekre távolságnak a NaN-t (Not a Number, nem szám) kap a program, vagyis ahol nem tudja meghatározni a távolságot, azt 0 távolságnak vagy végtelennek vegye. Praktikusabb a végtelent választani, mert ez esetben feketével jelenik meg a mélységi képen, míg ha 0-val helyettesíti, akkor lila lesz. Emellett beállíthatjuk, hogy a diszparitás képet szeretnénk-e, hogy interpolálja, vagyis közelítse. (A két képből először generál egy

diszparitás képet, ami megadja, hogy az ugyanahhoz a tárgyhoz tartozó pontok mennyire távol vannak egymástól a két képen. Ebből a képből generálható utána a mélységi kép.) Ezután választhatunk, hogy Semi Global Matching Algoritmust vagy a Block Matching algoritmust szeretnénk használni, illetve ezek tulajdonságait beállíthatjuk. Az általam használt beállításokat a LabView beépített mintaprogramjából vettem, mert a sima alapbeállításoknál nem jelent meg rendesen a kép. Ezután indítsuk a programot, ami rögtön feldob egy ablakot, ahol



5.3.1. ábra - A sztereo rendszer használata - Beállítások

kiválaszthatjuk az előzőleg elmentett kalibrációs fájlt. Ha ez megtörtént, át is mehetünk a második fülre, ahol a tényleges mérést végezhetjük.



5.3.2. ábra - A sztereo rendszer használata - Eredmény

Ezen a lapon láthatjuk bal fönt a két kamera képét, bal lent az ezekből alkotott diszparitás képet (disparity image) majd az ebből generált tényleges mélységi képet. A mélységi kép színeinek skálázása az általunk megadott határok között zajlik, a lila felel meg a minimális távolságnak majd innen a piros-sárga-zöld-kék színsoron át nő a távolság egészen a maximális távolságot jelentő feketéig. Ha valami ezen távolságon kívül esik, vagy a program nem tudja kiszámolni a mélységi koordinátáját, akkor NaN (Not a Number) kerül a távolság helyére, ami a fenti ábrán feketével jelenik meg. Ha a Live kép kapcsolót kikapcsoljuk, akkor megmerevíti az aktuális képet. Ez kimondottan akkor hasznos, ha a konkrét mélységet szeretnénk megtudni. Ehhez a mélységi kép fölött kell mozgatni az egeret, és a kép fölé mindig kiírja az aktuális ponthoz tartozó mélységet centiméterben.

A program működése és a Blokk Diagram

A programot megint egy hatalmas Flat Sequence Structure foglalja keretbe. Az első képkockában egy while ciklus várja, hogy a felhasználó kiválassza a kalibrálás fájlt. Ha ezt elrontja, bezárja az ablakot vagy egyéb hiba lép föl, akkor a még kétszer újra kéri őt a gép, harmadik próbálkozás után azonban leállítja a vi-t.

Ezután a második kockában létrehozuk a stereo sessiont, betöltjük a kiválasztott config fájlt és létrehozuk a képeket, majd továbblépünk a harmadik képkockára. Itt végzi egy while ciklus

a tényleges mérést. Először lekérjük a kamera képét, majd az IMAQ Stereo Correspondence vi-t, amiből külön van a két féle egyezési algoritmusnak. Ezután meghívjuk (ha a felhasználó kérte) az IMAQ Interpolate Disparity Image-t. Ezzel már meg is van a diszparitás képünk. Ebből mélységi képet a IMAQ Get Depth Image From Stereo vi-jal csinálhatunk. Amit így kapunk, az egy két dimenziós double típusú tömb, ami mindegyik pixelértékhez a hozzátartozó mélységértéket tárolja. Ezt a legjobban egy Intensity Graph-on tudjuk ábrázolni. Ez minden értékhez egy színt rendel, ahogy a front panelen is látható.

6. Források, felhasznált irodalom

1. NI Vision for LabVIEW - User Manual, 2005.
2. Előadás diák Informatika II. tárgyból – Aradi Petra, 2013.
3. Egyetemi jegyzetek és előadás diák a Fotogrammetria és Térinformatika Tanszék (BME) oldaláról
4. A forums.ni.com oldalról K13m3n munkája
5. Mészáros Ádám - Surveyor SVS robot navigáció sztereó rekonstrukció segítségével, 2011. diplomamunka, Kecskeméti Főiskola