Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/2.

Drónok szabályzásainak vizsgálata képfeldolgozással

Dezső Dávid
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
dezso.david15@gmail.com

Sarvajcz Kornél
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
sarvajcz@eng.unideb.hu

Dr. habil. Husi Géza
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
husigeza@eng.unideb.hu

***Absztrakt*— A kutatás során egy képfeldolgozó szoftvert készítettünk, amelyet drónok repülését kiértékelő vizsgálatok során alkalmazható. Olvasható egy quadkopter szabályozásának bemutatása, a képfeldolgozási technika ismertetése, és a valós körülmények közötti vizsgálat során kapott eredmények.**

Kulcsszavak—szabályozás, képfeldolgozás, OpenCV, drón, quadkopter, pozíciótartás, Python

I. BEVEZETŐ

Az utóbbi években felgyorsult a távvezérlésű quadkopterek térhódítása, mind a hobbi mind a professzionális felhasználás terén. Kiemelkedő fejlődést mutatnak a felvételek készítésére és képalkotásra használt pilóta nélküli járművek.

Ilyenekkel foglalkozunk a Debreceni Egyetem Mechatronikai Tanszék Drón klubjában. Az ilyen felhasználású gépek általában nagyobbak, mint versenyzésre használt társaik, és fő szempontjuk a stabilitás. Vagyis helyzetük pontos tartása a lehető legkisebb ingadozással és elúszással. Ennek eléréséhez számos szenzorra és azok összehangolt együttműködésére van szükség.

A szabályzás beállítása aprólékos feladat és sok finomhangolást igényel. Adott változtatás után akkor lehetünk csak biztosak benne, hogy a jó irányba haladunk, ha repüléssel teszteljük a pozíciótartás minőségét. Annak megfigyelésére, viszont kizárólag a saját megfigyelésünket használhatjuk. A két teszt között eltelő hosszabb rövidebb idő, a megfelelő időjárás és alkalom kivárása, miatt nehéz megállapítani a stabilitásban előidézett javulás, avagy romlás mértékét.

A kutatás során az a cél, hogy erre a problémára egy megfelelő megoldást dolgozzunk ki. Szándékunkban áll egy olyan vizsgálat kidolgozása, amivel számadatokkal mérhető egy quadkopter pozíció tartásának stabilitása, így összehasonlíthatóvá válik saját korábbi verzióival, vagy éppen más drónokkal.

A tervünk egy olyan program készítése, amely egy drónról készült videó felvételt, képfeldolgozási algoritmusokat alkalmazva, dolgoz fel. A quadkoptert detektálja minden képkockán és útvonalát rögzíti a vizsgált időszakban. Ezt követően a mozgását elemzésre alkalmas formában megjeleníti.

A teljes vizsgálat a videó felvétel megfelelő elkészítéséből, a képfeldolgozó programmal való analizálásból, végül az eredmények kiértékeléséből áll majd

II. EGY QUADKOPTER ÉRZÉKELŐI ÉS SZABÁLYZÁSA

A. Érzékelők

Ahhoz, hogy egy multikopter stabilan a levegőben tudjon maradni és megfelelően manőverezni, szabályzásra van szükség. Egy ideális világban, ahol minden szerkezeti elem tökéletesen működik és nincsenek külső környezeti hatások szabályzásra sem lenne szükség. Viszont a való életben nincs két tökéletesen egyforma motor, vagy légcsavar. Ezért érzékelőkre van szükség, melyek „szemmel tartják” az UAV állapotát, elektronikára, mely a rendelkezésre álló adatok alapján vezérlik az aktuátorokat.
A repüléshez elengedhetetlen szenzorok az előző alfejezetben tárgyalt IMU-ban helyezkednek el. Egyike ezeknek a giroszkóp. A giroszkóp feladata annak a megállapítása, hogy az UAV a vízszintes helyzetétől mekkora szögben és milyen irányban tér el. Ehhez a perdületmegmaradás elvét használja. A mindennapi életben számos helyen használnak ilyen érzékelőket, a legszemléletesebb példa a hagyományos repülőgépek kezelő felületén látható műhorizont, melyet teljesen azonos okból használnak, mint a quadkopterekben. Két giroszkóp a horizontális és a vertikális tengelyek menti dőléseket is érzékeli, e három tengely körüli elfordulásra Magyarországon is az angol megnevezések használata az elterjedt:

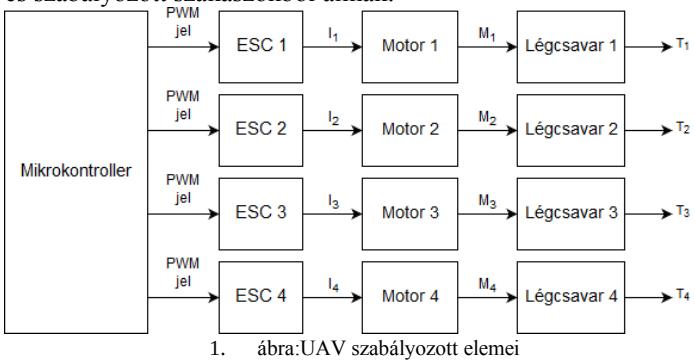
- Yaw = függőleges tengely körül elfordulás
- Roll = az előre menetirányra merőleges vertikális tengely körül elfordulás
- Pitch = az előre menetiránnyal párhuzamos tengely körül elfordulás

A perdületmegmaradás ellenére a giroszkópokban fellép az elfordulás hatására az úgy nevezett „drift”. Ez a forgási sík elmozdulása. Itt lép színre a mérőrendszer következő eleme, a gyorsulásmérő. Az erőhatás érzékével képes kompenzálni a giroszkópban jelentkező driftet. A gyorsulásmérő működési elvét tekintve azonos egy rugókhoz erősített test viselkedésével, a testet érő erőhatás azt elmozdítja, ezzel adott rugót nyomva, vagy húzva. Az eddig bemutatott két szenzor

szoros együttműködésben áll és adataik összegzésével lehet pontos információt elérni.
A helyzet megismerése után éppoly fontos a pozícióé. Abból is elsődlegesen a magasság. Ehhez egy barométer kerül integrálásra az IMU-ba. A barométer a légnyomás mérését végzi. Az, hogy egy multikopter milyen magasságban helyezkedik el meghatározza, hogy mekkora levegő oszlop nehezedik rá. Természetesen a drónt körülvevő légnyomásra nagy hatással van a levegő hőmérséklete. Ezért van szükség egy hő szenzorra is a légnyomásmérő mellett. A kettő adataiból már kiszámítható, milyen magasságban van éppen a quadkopter.
A magasság megállapításánál maradva következzen egy szenzor, ami a középkategóriás és feletti profi drónok alapfelszereltségévé vált az elmúlt években. Ez pedig az ultrahangos távolság mérő. A quadkopter aljára helyezik függőlegesen lefelé nézve. Egy ilyen szenzor egy „triggernek" nevezett adóból küld ultrahang hullámot, amit az „echo" vevő érzékel miután az egy objektumról visszaverődik, az objektum esetünkben a talaj. A kibocsátott és a beérkező hullám késése jelenti mennyi időben telt az ultrahang hullámnak adott távolságot kétszer megtenni. A hullám haladási sebesség ismeretében, ez 20 celsius fokon 343,2 m/s, kiszámítható a magasság. Persze az ultrahang hullám nem képes bármekkora távolságot megtenni. Így elsősorban a fel és leszállásnál, valamint a földközeli repüléseknél használható a légnyomásmérő kiegészítésére.
A teljesen autonóm pozíciótartáshoz, és repüléshez szükséges egy GPS vevő. Ez a szenzor a föld körül keringő műholdak által sugárzott időkódok és pályaadatok, és saját órája segítségével „háromszögeléssel" határozza meg a pontos koordinátáit, ehhez minimum 4 műhold érzékelésére van szüksége. [3]

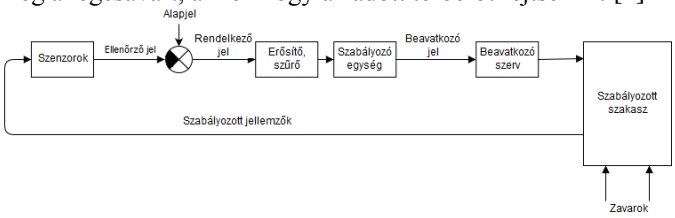
B. Szabályozás

A szabályzáshoz használt érzékelők ismertetése után ejtsünk néhány szót magáról a szabályzásról. Szabályzásra azért van szükség, hogy az UAV stabilan képes legyen megtartani az állapotát repülés közben. A stabilitás itt azt jelenti, hogy a környezeti hatások, és saját elemei tökéletlenségének ellenére is egyensúlyi állapotban marad. Egy ilyen rendszer szabályozó és szabályozott szakaszokból állnak.



1. ábra:UAV szabályozott elemei

Az 1. ábrán látható egy quadkopter szabályozott elemei. A mikrokontroller a különböző kitöltési tényezőjű PWM jelek küldésével tesz eleget a szabályozó szakaszból érkező parancsoknak. A motor vezérlő elektronikája alakítja át a PWM jelet a BLDC motor működtetéséhez szükséges áramerősséggé. A motor a szükséges nyomatékkal forgatja meg a légcsavart, ahhoz hogy az adott tolóerőt fejtsen ki. [2]



2. ábra: Szabályozó szakasz

A quadkopterekben PID szabályozó egységekkel történik a beavatkozó szerv, vagyis a mikrokontroller, számára szükséges parancs létrehozása. A szenzorok jelének egyesítése után egy különbségképző szerv összeveti az alapjellel, mely a robotpilóta, vagy a távvezérlőt kezelő személy által küldött parancs. A szükséges erősítés és szűrést követően ér el a PID egységhez. Egy ilyen egység három tagból áll: A P, arányos, tag kimeneti jele arányos a bemeneti jelével. Az I, integráló, tag, amely kimeneti jele a bemeneti jel integráljával arányos. Végül a D, differenciáló, tag, ennek a kimenő jele a bemenőjel változási sebességével arányos.
A legtöbb quadkopter központi computerében előre van programozva egy önbeállítás a szabályzáshoz, de előfordul, hogy sajátkezű beállítással jobb eredményt érünk el. Az egyéni, akárcsak a beépített, kompenzáló paraméter beállítás a Ziegler-Nichols módszerrel történik. Ekkor a hurokerősítést addig növeljük, amíg állandósult lengést nem kapunk. Ez a drón esetében lebegő helyzetben folyamatos remegésként jelentkezik. Ekkor meghatározható a kritikus körerősítéshez tartozó periódus idő. Ezekkel az adatokkal meghatározható mindhárom tag értéke, mérésekkel kialakított összefüggések használatával. [3]

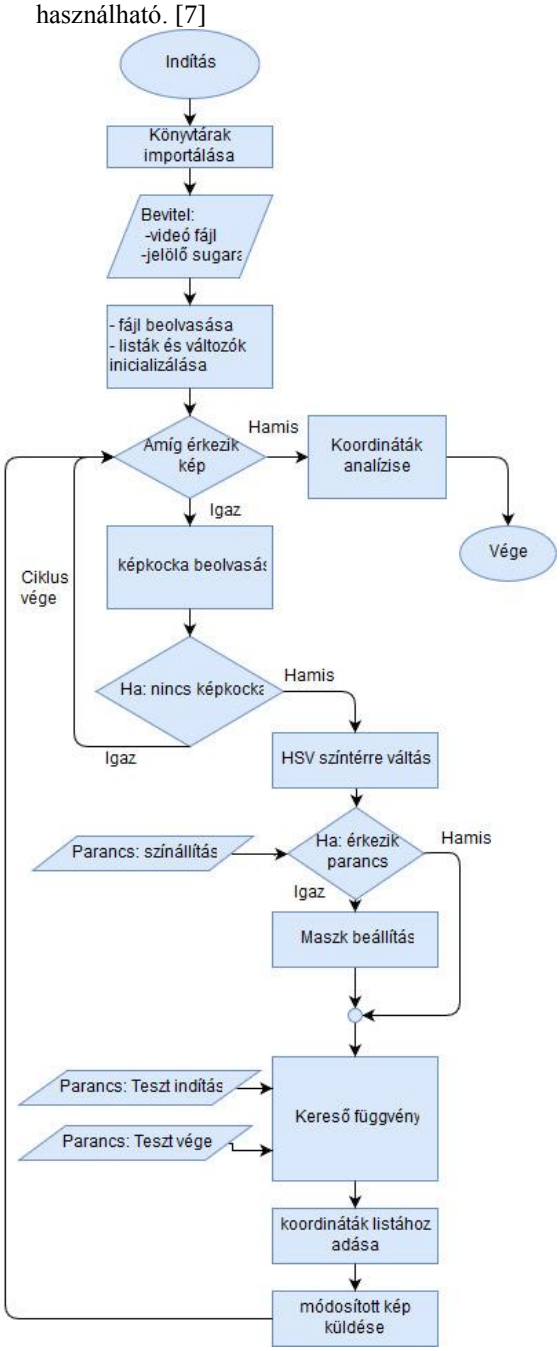
III. OBJEKTUM KÖVETŐ PROGRAM

A. Könyvtárak importálása

A program működéséhez szükséges könyvtárak listája és rövid jellemzése:

- **CV2:** Az OpenCV (Open Source Computer Vision) könyvtár képfeldolgozás és gépi látás alkalmazására szolgáltat több ezer algoritmust. [4]
- **Numpy**: Az OpenCV mellé elengedhetetlen algoritmus csomag. A Python tudományos számításokra alkalmas alapvető könyvtára. Tartalmaz több dimenziós tömb elemek létrehozására függvényeket. Eszközöket C/C++ scriptek integrálására. Használható lineáris algebra, Fourier transzformáció számítására és random szám generálásra. [1]
- **Matplotlib**: A Python két dimenziós rajzoló könyvtára. Segítségével prezentálásra és kiértékelésre alkalmas diagramok hozhatók létre. Interaktív környezet platformok közötti adatközléshez. [2]

- **Imutils**: Adrian Rosebrock által az OpenCV-hez készített és szabadon felhasználhatóvá tett könyvtár, amely képek méretezéséhez, alakításához használható. [7]



3. ábra: A kész program egyszerűsített folyamatábrája

B. *Adatok Beolvasása*

Egy Python program elindításakor rögtön felugrik egy kezelő ablak. Az indítást követően ebben a következő üzenet jelenik meg:
„Insert name of the video file:”
Vagyis a programnak a vizsgálni kívánt felvétel nevére és kiterjesztésére van szüksége. Abban az esetben, ha a fájl nem azonos mappában van a programmal, akkor a nevét az elérési útjával együtt szükséges megadni. Mivel az OpenCV-nek elsődlegesen a valós idejű képfeldolgozás a fő csapásiránya, így természetesen erre is van lehetőség. Ha folyamatos közvetítés szeretnénk feldolgozni, a videó fájl neve helyére a kamera sorszámát kell megadni, vagyis egy csatlakoztatott felvevő esetén „0”-t. Miután a fájlnév beolvasása megtörténik, egy további kezdő változó megadása is szükséges, ez pedig az objektum jelölő kör sugara milliméterben:
„Radius of the circle (in mm):”
Ehhez az inputhoz megadtunk egy alapértelmezett értéket, ami 25 *mm*. Ennek elküldésével a feldolgozás kezdetét is veszi, és a kezelő ablakban megjelenik a következő üzenet:
„Press 'C' to set the color (esc if done) S' to start test and 'Q' to quit.”
A 3. ábrán látható folyamatábra while ciklusában három bemenet van, ezek a szoftver kezelésére szolgálnak. A fenti üzenet erről nyújt felvilágosítást.

C. *Detektáló ciklus*

Miután megtörtént a videó fájl behívása, és a szükséges változók és a listák inicializálása a program belép a fő ciklusába. Itt történik meg az objektum detektálása, valamint a mozgás analízishez szükséges adatok rögzítése. Első lépésként a felvétel soron következő képkockájának behívása történik a „getframe” függvénnyel, ami a kép mellett egy boolean értéket is visszaküld arra vonatkozóan, hogy létezett a behívni kíván kocka. Ha üres képet kapott, akkor „Hamis” értéket kap a változó. Ennek a vizsgálata következik, így ha kifogy a képből a program akkor megszakítja a ciklust és a koordináta analízis részhez ugrik. Most a kép elő alakítása történik az „imutils” könyvtár segítségével lekicsinyíti, ezzel gyorsítva a feldolgozás folyamatát. Majd színteret vált a „cvtColor” függvény. RGB kódolásból átvált HSV (H=árnyalat, S=telítettség, V=világosság) színtérre. Ezzel a kódolással már meghatározhatóak a határok, amik között ha csak a jelölő marad akkor sikeres lesz a detektálás a színkereső függvényben. Az értékek meghatározása történik a most következő részben. Ekkor már a videó fájl képről képre történő megjelenítése zajlik, és a program arra vár, hogy a parancsok valamelyikét megkapja a kezelőtől. A „C” billentyű lenyomása küldi el a „colorset”, azaz színbeállítás parancsot. Ekkor a videó megáll adott képkockán. A beállításhoz egy új ablakot hoz létre a program, így már három ablakot látunk a pillanatnyi képkockát, a maszkot, valamint egy olyat melyben hat darab csúszka képviseli a színhatárok értékeit. Miután ezek megjelentek a program egy while ciklusba lép. Itt folyamatosan beolvasásra kerülnek a csúszkákon beállított pozíciók. Ezeket felhasználva létrehoz egy új maszkot, amit az előző ablakában frissít. A határ keresést érdemes a színárnyalat beállításával kezdeni, mivel az csak kis mértékben változhat, valamint a homogén jelölőnek köszönhetően pontosan, azaz szűk határok között, megtalálható. Ezt követően a színtelitettség határainak összehúzásával érdemes finomítani a képen. A végső a

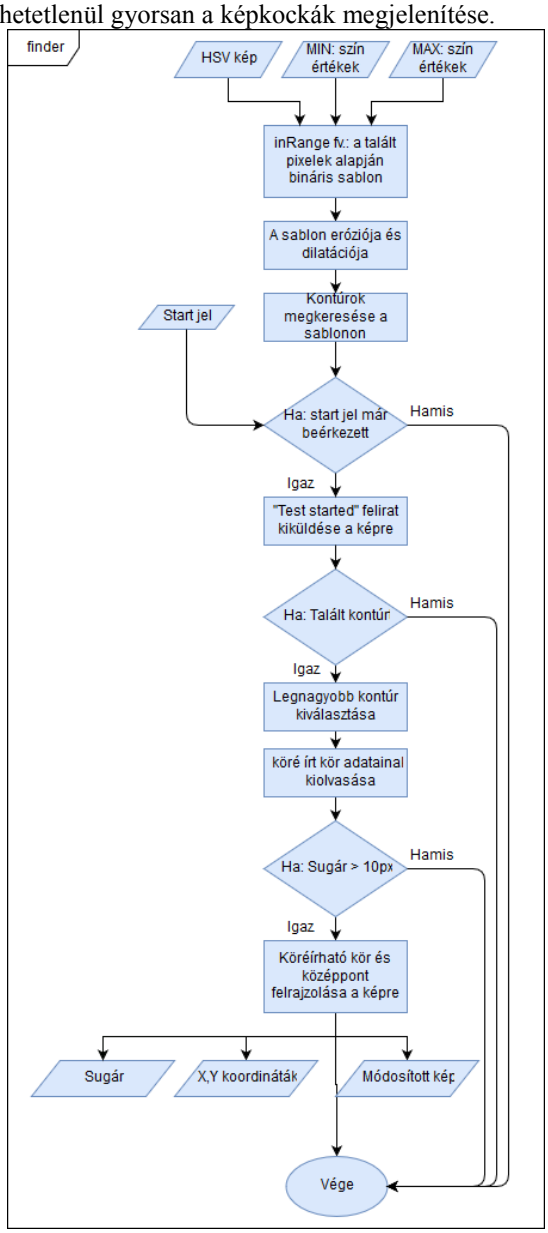
világosság beállítása, ebben az esetben nem szerencsés csak az adott képkockához állítani mivel a drón mozgásával változhat a megvilágítottság, és ha kívül esik a határon, akkor hibás beolvasások fognak történni. Ha végeztünk a határok megszabásával az „ESC” billentyűt leütve elfogadjuk a jelenlegi HSV értékeket és a script fő ciklusa folytatódik tovább. Az általunk írt színkereső („finder”) függvény behívása történik ezután, ami detektálja a képen a jelölőt, módosítja a kijelzett képet és visszaküldi az X és Y koordinátákat, valamint a jelölő sugarát képpontokban.

D. Színkereső függvény

Az előző fejezetben már említett „finder” függvény adja a szoftver gyökerét. Behívásakor három argumentum megadására van szükség:

- hsv: Az új színtérbe helyezett kép.
- MIN: A „colorset” csúszkákon beállított minimum határok.
- MAX: A „colorset” csúszkákon beállított maximum határok.

Ezekre mind az első lépésnél van szükség. Ahogyan a maszk beállításánál is történt itt is az „inRange” függvényt használja a program a jelölő sablon képre történő kiemeléséhez. Ez minden pixelt végig vizsgál, és amelyiknek a színértékei mind a meghatározott határokon belül helyezkednek el. Az azoknak megfelelő pixeleknek „Magas” értéket ad a sablonon, minden más logikai „0” értéket kap, így a sablon maga egy bináris tömbként értelmezhető. A következő feladatunk a kontúrok megkeresése lenne, azonban ebben az esetben a program működése nem lenne kifogástalan, többször is hibára futna, ezzel lehetetlenné téve a teszt elvégzését. Mégpedig azért, mert sablon képen lévő maszk széle egyenetlen és összességében is túl zajos az elszórtan előforduló „Magas” pixelektől. A probléma megoldására szolgálnak a morfológiai átalakítások. A morfológia jelentése alaktan, A képfeldolgozásban több összefüggő képalakítási folyamat gyűjtőneve. A mi esetünkben az erodációt és a dilatációt alkalmazza a szoftver. Az erodáció során elcsúsztatja a sablont saját maga alatt és csak az a pixel marad „1” ami alatt is magas érték található. Így eltűnnek az apró zajok a képről, ugyanakkor a számunkra fontos kontúr is kisebb lesz. Bár a program a lecsökkent méretű maszkot is gond nélkül megtalálná az analízis későbbi folyamatához begyűjtött rádiusz értékek módosulnának, ezzel rontva a teszt végeredményének pontosságát. Ezért van szükség a dilatációra is. Ami ugyanennek az inverzét teszi, vagyis minden pixel magas értéket kap, aminek elcsúsztatott vagy eredeti értéke „1”. Itt visszakapja eredeti méretét a keresett kontúr, de az eltűntetett zajok nem jönnek vissza. Látható, hogy a két folyamat azonos, a különbségük abban rejlik, hogy az erodáció a logikai „és” a dilatáció pedig a logikai „vagy” alapján cselekszik. Ezt követően a sablon már alkalmas a kontúrok megkeresésére. Ez azt jelenti, hogy a „0”-„1” határok keresésével határozza meg a látható formákat. Mindezt a CV „findContour” függvénye teszi meg. Létrehoz egy tömböt az összes talált kontúr adataiból, a feladat csupán a legnagyobb kiválasztása ezek közül, elvégre a határ beállítás úgy történt, hogy lehetőleg csak a jelölő jelenjen meg rajta. A szoftver eddig a pontig minden képkockán lefut. Azért alakítottuk így mert ezek az átalakítások a leg összetettebb műveletek, amelyek a legtöbb időt igénylik. Éppen ezért, ha a teszt indítása megtörténik nem jelentkezik feltűnő lassulás a feldolgozásban, és a kezdetéig pedig nem történik követhetetlenül gyorsan a képkockák megjelenítése.



4. ábra: Színkereső függvény

A teszt indítása a folyamatábrán (4. ábra) látható start jel küldésével kezdődik, ahogyan a már említett magyarázó üzenetből kiderül ez az „S” billentyű lenyomását jelenti. Elsődleges visszajelzésként az egy sárga színű „TEST STARTED” felirat jelenik meg az ablak bal felső sarkában. Majd ellenőrzésre kerül, hogy talált-e kontúrt. Ha igen akkor használatra az OpenCV „minEnclosingCircle” algoritmusa. A legnagyobb kontúr argumentumként történő megadásával

három értéket küld vissza, mindet a kontúr köré írható legkisebb körről. Ezek sorrendben: a kör középpontjának X koordinátája, az Y koordinátája, végül a sugara. Ha a rádiusz meghalad egy minimum értéket akkor visszajelzésként a képre kirajzolásra kerül a kör és a középpont is. Ezzel a függvény végére is értünk. Zárásként visszaküldi a koordináta és sugár értékeket.

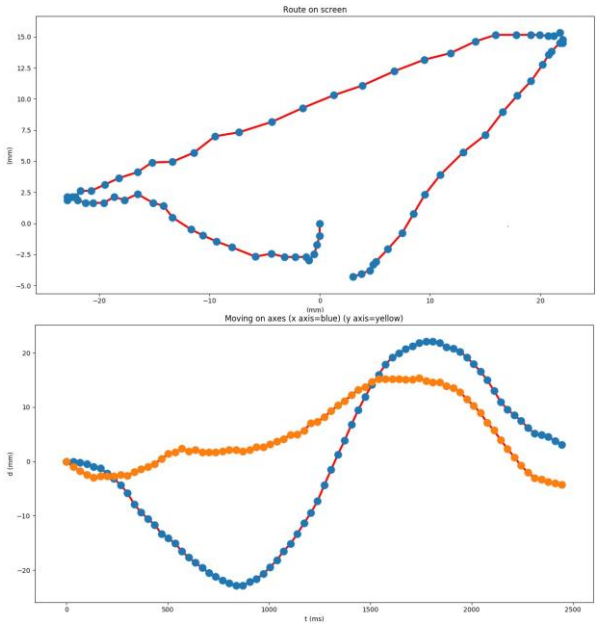
E. Koordináták analízise

A fő ciklus három darab listát készít el az Y-, az X-koordinátákét és a sugarakét. Ezek minden képkockához tartalmaznak egy elemet, a teszt indítása előtt „0” kerül beírásra, éppen ezért szükséges ezek kiszűrése. Erre a feladatra egy egyszerű függvényt definiáltunk a script rövidítése céljából. Az eredeti lista a képkoordinátákat tartalmazza, vagyis az objektum helyzetét a képen. Első átalakításként egy relatív elmozdulás lista kidolgozása történik. A szűrésen átesett koordináta listák első elemét nevezi ki az origónak, az összes többit lecseréli ezen ponttól való távolságukra, különbségükre. Ez még mindig csak a százalékos eltérés összehasonlítására alkalmas, az általános használathoz a valós elmozdulás megállapításra is szükség van. Ennek elérésére szolgál a rádiuszokat tartalmazó lista. A szoftver ugyanis ismeri a jelölő valós sugarát, hiszen a feldolgozás kezdetén a felhasználó ezt megadta. A következő összefüggéssel számítja ki a program adott elmozdulás a képen mekkora valós elmozdulást jelent a látható tengelyeken:

$$\frac{Leolvasott\ sugár}{Valós\ sugár} = X \tag{1}$$

$$X = Milliméter - képpont\ váltószám \tag{2}$$

$$\frac{Leolvasott\ elmozdulás}{X} = Valós\ elmozdulás \tag{3}$$



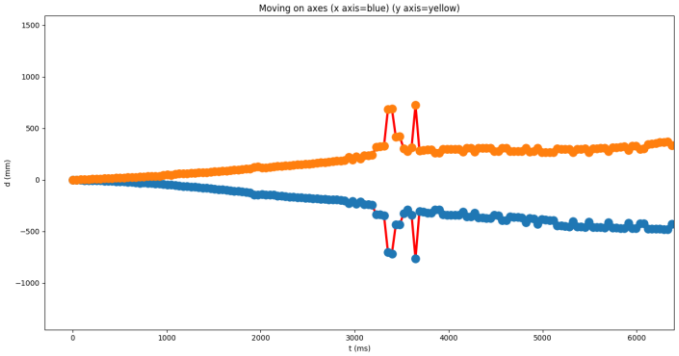
5. ábra: Vizsgálat végi diagramok

A felső ábra a képfeldolgozás eredményének első megjelenítését mutatja. Az X és az Y tengelyek visszaadják a teljes mozgás képét. Mindkét tengelyt milliméterben jelöli. Ez a diagram inkább a követés sikerességét mutatja mintsem a kiértékeléshez szükséges. A mozgás kiértékelésére szolgáló diagramon két függvény kerül megjelenítésre. A függőleges „d” tengely elmozdulást mutat, a vízszintes „t” tengely pedig az időt. Amikor a program létrehozza a diagram nagyítható akár tized milliméteres és milliszekundumos pontossággal leolvasható. A kész program rendszerkövetelményei: 1 GHz sebességre képes processzor, 80 MB RAM memória.

IV. POZÍCIÓTARTÁS VIZSGÁLATOK

A. Phantom 1 teszt

A tesztet eredetileg labor körülményekre gondoltuk ki, vagyis megfelelő gps vétel, jó megvilágítás, és szélcsend. Ezek nem álltak rendelkezésre, de így megfigyelhető milyen hatásokat eredményez bármelyik hiánya. Ennek a vizsgálatnak az időpontjában állandó 7 km/h-s szél és erős alkalmankénti 15 km/h-s széllökésekkel kellett megküzdenie repülés közben az eszköznek, a levegő hőmérséklete 12°C volt. Abban az esetben is erős volt a légmozgás viszont mind iránya mind nagysága állandó volt. A drón nem volt képes a pozícióját megtartani, de láthatóan a szél ellen dolgozott, szabályozás nélkül irányítás hiányában még a váz vízszintes közeli megtartása sem valósult volna meg. Egy folyamatos enyhén gyorsuló elúszás figyelhető meg. 3,5 másodperc környékén ebben a mérésben is történt pár kiugrás, de ennek ellenére értékelhető. Ezt követően mér nem olyan simák a függvények, mint előtte, erre is van magyarázat, a quadkopter folyamatosan távolodott ekkor már több mint öt méterre járt a kamerától. A mozgás kiértékelését az első három másodpercre korlátoztuk.



6. ábra: Phantom mérés

1. táblázat: Phantom mérés

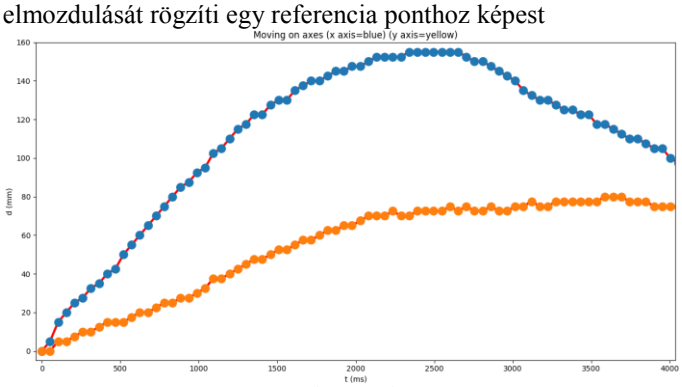
t [s]	0,5	1	1,5	2	2,5	3
d_X [mm]	-17,4	-49,9	-86,8	-136,7	-178	-208,3
v_X [m/s]	-0,0348	-0,065	-0,0738	-0,0998	-0,0826	-0,0606
a_X[m/s^2]	0,0696	0,0604	0,0176	0,05	-0,0344	-0,044
d_Y [mm]	21,7	49,9	78,1	117,2	154,1	200
v_Y [m/s]	0,0434	0,0564	0,0564	0,0782	0,0738	0,0918
a_Y [m/s^2]	0,0868	0,026	0	0,0436	-0,0088	0,036

Az „X” tengelyen történő vízszintes mozgás negatív irányú ez azt jelenti, hogy a kép bal széle felé úszott, ami a drón

szemszögéből jobbra húzást jelent, ez megegyezik a szélirránnyal. A sebesség két másodpercig változó mértékben gyorsult, ezután enyhén lassulni kezdett, de a teljes diagramon látszódik, hogy a sebesség iránya nem változott.
A vertikális tengely menti mozgás folyamatos emelkedést mutat, ami két kis kilengést leszámítva szintén folyamatosan gyorsul. Mindkét vizsgált tengelyen megközelítőleg 20 centiméterre került a kezdeti ponttól három másodperc alatt a drón.

B. Inspire 1 teszt

Az elkészült program egy fordított megközelítéssel történő használatát próbáltam ki a DJI Inspire 1-es drónnal készült vizsgálat során. Ez esetben a jelölő nem a vázra került rögzítésre, hanem egy stabil pontra, konkrétan egy csatornafedélre a földön. Így a program a videó készítő eszköz elmozdulását rögzíti egy referencia ponthoz képest



7. ábra: Inspire mérés

2. táblázat: Inspire mérés

t[ms]	400	800	1200	1600	2000	2400	2800	3200	3600	4000
d_X [mm]	39	77,2	110,5	133,8	147,5	155	150	130	117	100
v_X [m/s]	0,195	0,191	0,167	0,117	0,069	0,038	-0,03	-0,1	-0,07	-0,09
a_X [m/s^2]	0,98	-0,02	-0,12	-0,25	-0,24	-0,16	-0,31	-0,38	0,18	-0,1
d_Y [mm]	14,5	25,2	40,3	54,6	66,3	72,6	72,6	75	80	75
v_Y [m/s]	0,073	0,054	0,076	0,072	0,059	0,032	0	0,012	0,025	-0,03
a_Y [m/s^2]	0,363	-0,095	0,11	-0,02	-0,07	-0,14	-0,16	0,06	0,065	-0,25

A vizsgálat ideje alatt szélcsend volt és körülbelül 12°C. Egyetlen hiba tényező lehetett az, hogy sűrűn beépített környezet vette körül, ami a GPS vételt zavarhatta. Egy lassú csúszás itt is megfigyelhető, ugyanakkor az is, hogy ezt a szenzorok érzékelték. Az „X” tengely mutatja a haladási irányra merőleges mozgást. A kezdeti ponttól jobbra indult meg 0,195 m/s sebességgel. Erre rögtön reagált a szabályzás és finoman lassítani kezdte és mire elérte a maximális 15,5 cm-es kitérést el is indult visszafelé. Nagyon hasonló tendencia figyelhető meg az „Y” tengely mentén. Előre indul el, a másik tengelytől lassabb kezdő sebességgel. A válasz lassulás is kisebb mértékű, és bár 2,8 másodpercnél megáll a maximális kitérését 3,6 másodperc után éri el, majd korrigál visszafelé.

V. ÖSSZEFOGLALÁS

A kutatómunka készítése során sikerült olyan szoftvert létrehoznunk, mely szín detektálási technikát alkalmazva képes egy megjelölt objektumot lekövetni a róla készült videó felvételen. Majd kiegészítettük olyan tulajdonságokkal, melyek alkalmassá teszik egy quadkopter pozíció tartó mód közbeni stabilitásának meghatározására irányuló vizsgálatban való használatra. Ezen tulajdonságok közé tartozik a követett szín határainak egyéni beállítása, a felvétel vizsgált időtartamának parancsra történő indítása és zárása, a valós elmozdulások meghatározása, valamint a vizsgálat eredményének kiértékelhető formában való megjelenítése. Mindezek teljesülése után megtörtént a valós körülmények közti teszt. Két kereskedelmi forgalomban kapható UAV használatával történtek meg a mérések. Megfelelő állapotok mellett a szoftver alkalmazható quadkopterek fejlesztése alatt a szabályzásban, vagy hardverben eszközölt változtatások hatásának ellenőrzésére.

KÖSZÖNETNYILVÁNÍTÁS

A publikáció/prezentáció/poszter elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

HIVATKOZÁSOK

[1] „Numpy,” Numpy, 2017. [Online]. Available: http://www.numpy.org/. [Hozzáférés dátuma: 30 09 2017].

[2] „Matplotlib,” Matplotlib, 13 10 2017. [Online]. Available: https://matplotlib.org/. [Hozzáférés dátuma: 02 11 2017].

[3] „hobbielektronika.hu,” 15 05 2015. [Online]. Available: https://www.hobbielektronika.hu/cikkek/multicopterek_nullarol_az_kig_i.html?pg=1. [Hozzáférés dátuma: 03 11 2017].

[4] „Opencv bemutatás,” OpenCV, 2017. [Online]. Available: https://opencv.org/about.html. [Hozzáférés dátuma: 15 10 2017].

[5] Dr. Tóth János, Szerző, *Mérés- és Irányítástechnika II.*. [Performance]. Debreceni Egyetem, Műszaki kar, Mechatronikai Tanszék, 2012.

[6] Dr. Tóth János, Szerző, *Mérés- és irányítéstechnika I.*. [Performance]. Debreceni Egyetem, Műszaki kar, Mechatronikai Tanszék, 2012.

[7] A. Rosebrock, „Pyimagesearch,” 2017. [Online]. Available: https://www.pyimagesearch.com/. [Hozzáférés dátuma: 30 09 2017]