[2018/11/02 v3.12 biblatex localization (PK/MW)]

[2018/11/02 v3.12 biblatex localization (PK/MW)]

Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

SZAKDOLGOZAT

Kamera alapú beltéri navigáció többrotoros pilóta nélküli repülőgépekkel

Mokos Mihály
Mérnök Informatikus BSc

2019

Témavezető: Dr. Zsedrovits Tamás

Nyilatkozat

Alulírott Mokos Mihály, a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának hallgatója kijelentem, hogy ezt a szakdolgozatot/diplomamunkát meg nem engedett segítség nélkül, saját magam készítettem, és a szakdolgozatban/diplomamunkában csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva / a dolgozat nyelvétől eltérő nyelvből fordítva, más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem. Ezt a szakdolgozatot/diplomamunkát más képzésen nem nyújtottam be.

Mokos Mihály

Kivonat

A dolgozat a kamera alapú beltéri navigációs rendszerek különböző autonóm megvalósításait tárgyalja pilóta nélküli repülőgépek segítségével. Az elméleti háttér és a felhasznált eszközök bemutatása mellett, a gyakorlatban való alkalmazást, valamint tesztelést elemzi.

A kutatás során felhasznált hardveres és szoftveres eszközök egy egységet képeznek, melyek egy irányvonalat mutatnak az olvasónak navigációs rendszerek tervezéséhez. A gyakorlatban alkalmazott eszközök ismertetése mellett, egyéb más megoldási módszereket is említ, melyek támogatott részei az egységnek, így könnyen integrálhatóak a jelen navigációs rendszerekbe.

A dolgozat, két eltérő paraméterekkel rendelkező pilóta nélküli repülőgéppel foglalkozik, melyeken a tesztek is eltérőek. Az tesztelés során felhasznált eljárások a két repülőgép tulajdonságaihoz igazodnak, így törekedve a teljeskörű elemzésre és optimális felhasználásra. Az eljárások kamera alapú adatok alapján működnek, valamint szenzoraik függvényében IMU adatok igénybevételével.

Az első teszt a Parrot cég által gyártott AR. Drone 2.0 elnevezésű repülőgépen történt. Ez egy beltéri felhasználásra készült, viszonylag nagy méretű drón, mely két kamerával van felszerelve, az első és alsó részén. Az első kamera magas minőségű képet biztosít a felhasználójának. Alsó részén található egy ultrahangos távolságmérő és egy légnyomásmérő szenzor, továbbá egy 3 tengelyű giroszkóp, egy 3 tengelyű gyorsulásmérő szenzor, valamint egy magnetométer. Az eszköz 57 centiméteres hosszából és ugyanekkora szélességéből kifolyólag, akár kültéri tesztelésre is alkalmas.

A dolgozat első részében egy szövegfelismerő eljárás bemutatása és tesztelése valósul meg, az AR. Drone 2.0 eszköz felhasználásával. A CAPTCHA megoldó algoritmusok logikája alapján készült, így a repülőgép instabilitásából adódó, elmosódott képek esetén

is használható. A tesztelése csak részben sikeres, ezért további fejlesztéseket igényel. Ezen továbbfejlesztési lehetőségek a dolgozatban bemutatásra kerülnek.

A második teszt a Ryze cég által gyártott Tello Edu elnevezésű repülőgépen történt. A beltéri alkalmazásra optimalizált, mindössze 17,8 centiméteres hosszából és 16,8 centiméteres szélességből adódóan, kiválóan navigálható. Alsó részén található egy lézeres távolságmérő és egy légnyomásmérő szenzor, továbbá egy 3 tengelyű giroszkóp, egy 3 tengelyű gyorsulásmérő szenzor, valamint egy magnetométer. Az 5 megapixeles kamerája elektronikus képstabilizátorral van ellátva, így alkalmas magas minőségű képek biztosítására.

A dolgozat második részében egy QR-kód detekciós eljárás bemutatása és tesztelése valósul meg, a Tello Edu eszköz felhasználásával.

A kutatás során használt eszközök egyike az OptiTrack cég által forgalmazott kamerarendszer, mely mozgáskövetésre alkalmas. A tesztek alatt ez biztosítja a repülőgépek pontos helyét, a kamerarendszerhez tartozó fényvisszaverő markerek segítségével melyek a repülőn helyezkednek el.

A Robot Operating System egy nyílt forráskódú keretrendszer robotikai alkalmazásokhoz, mely számos hasznos modult és könyvtárat tartalmaz hasonló kutatásokhoz. A dolgozatban ezen rendszer kerül bemutatásra, továbbá ezen keresztül valósul meg a kapcsolat a repülőgépek és a kamerarendszer között. Az autonóm vezérlést Python programozási nyelven implementált program szemlélteti, a ROS rendszer felhasználásával. A program egy ROS csomagként jelenik meg a dolgozatban.

Az eljárások tesztelése egy akadálypályán való autonóm átrepülés formájában valósul meg. A pálya teljes területe az OptiTrack kamerarendszer által megfigyelt háromdimenziós tér. Az akadályok elkerülésére vonatkozó információt szöveges adatok, és QR-kódok hordozzák, melyek a különböző akadályokon helyezkednek el.

Abstract

The thesis discusses various autonomous implementations of camera-based navigation systems using unmanned aerial vehicles. In addition to presenting the theoretical background and the devices that have been used, it analyzes application and testing in practice.

The hardware and software tools that have been used in the research, form a single unit that provides the reader with a framework for designing navigation systems. Besides to describe the utilized implements in practice, there are mentioned other solutions, that are supported as part of the unit, so they can be easily integrated into the current navigation systems.

The dissertation analyses two different tests with two unmanned airplanes accompanied by different parameters. The procedures during the experiments adjusted to the characteristics of the two aircraft, aiming for full analyses and optimal usage. The methods work on camera-based data that are depending on their sensors using IMU data.

Testing began on an AR. Drone 2.0 aircraft manufactured by Parrot. This is a relatively large-sized indoor drone, equipped with two cameras on the front and bottom. The front camera provides the user with a high quality image. An ultrasonic rangefinder and an air pressure sensor have been placed on the bottom of the drone, moreover a 3-axis gyroscope, a 3-axis accelerometer and a magnetometer. Due to its length and width of 57 centimeters, the device is also suitable for outdoor testing.

The first part of the thesis introduces and tests an optical character recognition method, using the AR. Drone 2.0. The problem caused by instability of the aircraft instability can be solved with the logic based on the CAPTCHA resolving algorithms, so blurry images can also be utilized. This partially successful test requires further development. These possibilities for further development are presented in the dissertation.

The second test took place on a Tello Edu aircraft manufactured by Ryze. It is optimized for indoor usage, due to its 17,8 centimeters in length and 16,8 centimeters in width, it has excellent navigation. An laser rangefinder and an air pressure sensor have been placed on the bottom of the drone, moreover a 3-axis gyroscope, a 3-axis accelerometer and a magnetometer.The 5 megapixel camera is equipped with an electronic image stabilizer, accordingly it can deliver high quality images.

In the second part of the thesis, a QR code detection method is introduced and tested using the Tello Edu device.

One of the devices that have been used in the research is the camera system, marketed by OptiTrack, which is capable of motion tracking. During the tests, this ensures the exact location of the aircraft, with the help of reflective markers belonging to the camera system which are located on the aircraft.

The Robot Operating System is an open source framework for robotics applications that contains many useful modules and libraries for similar research. In the thesis, this system is introduced, and through this the connection between the aircraft and the camera system is realized. Autonomous control is illustrated by a program implemented in the Python programming language using the ROS system. The program appears as a ROS package in the dissertation.

The procedures are tested in the form of an autonomous flyover on an obstacle course. The entire track is a three-dimensional space observed by the OptiTrack camera system. The information on avoiding obstacles are as well represented by textual data as QR codes, located on the various obstacles.

Tartalomjegyzék

1. fejezet

Bevezetés

A kutatásom célja, hogy megvalósítsak két eltérő működésű, kamera alapú beltéri navigációs rendszert, melyek könnyen továbbfejleszthetőek és kellően univerzálisak a sokoldalú felhasználáshoz. Az előző évben már foglalkoztam ezzel a témával a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának UAV laboratóriumában. Az akkori munkám kellő tapasztalatot és inspirációt adott, ezzel előkészítve ezt a kutatást. Több hasonló projektet tanulmányoztam és arra a következtetésre jutottam, hogy ezen területen is, mint ahogy az informatika szinte minden területén számos kiaknázatlan lehetőség van a további fejlesztésekre.

A célkitűzéseim eleinte elrugaszkodottak voltak a teljesíthető feladatokhoz képest. Az interneten található cikkek és kutatások, melyek olvasva könnyűnek és kivitelezhetőnek tüntek, kis gondolkodás és fejlesztés után lehetetlenné váltak. Így kezdett körvonalazódni bennem, hogy az általam is elérhető források, képességeim, valamint a rendelkezésre álló időkeret milyen feladatok megoldását teszik lehetővé. Ezek alapján, próbáltam a további fejlesztések megalapozására, és a gyakorlatban való alkalmazásuk megteremtésére törekedni.

1.1. Kamera alapú navigáció

A képelemzés és a kamera alapú navigáció az egyik leggyorsabban fejlődő mező az informatika világában. Számtalan alkalmazási területei vannak, mind hétköznapi, mind pedig ipari szinten is. Napjainkban a képi és egyéb szenzorok adatainak feldolgozására egyre újabb és újabb algoritmusok jelennek meg. A kapott információ segítségével különböző számításokat végezhetünk, melyek felhasználásával automatizálási lehetőségek nyílnak,

például útvonalak optimalizálására, statikus helyzetben lévő tárgyak és dinamikus objektumok elkerülésére használható.

Egy praktikus és sok egyszerűstési lehetőséget teremtő funkció a gesztusvezérlés. Ez az új technológia lehetőséget ad arra, hogy különböző mozgások alapján tudjunk utasításokat adni egy rendszernek. Jó példa erre a BMW 530i nevű modelljében található HoloActive Touch elnevezésű koncepció, ahol a műszerfal előtti teret egy kamera érzékeli és a vezető a kezének, vagy akár újhegyeinek mozgatásával adhat parancsokat az autó egyes szolgáltatásaira vonatkozóan. [1] A pilóta nélküli repülőgépek esetében ez a módszer kiterjeszthető például egy tenyér követésére, vagy akár a kézfej forgatásával való dőlésszög változtatására is. [2]

1.2. Pilóta nélküli repülőgépek

A pilóta nélküli repülőgép elnevezés az angol Unmanned Aerial Vehicle szavakból ered, de ismertebben UAV rövidített formában használják világszerte. Ezen eszközök az ipar szinte minden területén megjelentek. Legtöbbet mégis a fegyverkezési és katonai szférában találkozhattunk vele. Mint ahogy a számítástechnikai és informatikai tudományágak megteremtője az első és második világháború volt, úgy a pilóta nélküli repülőgépek fejlődését és elterjedését is a katonai alkalmazás jelentette. Ezeket az eszközöket a hadászat már az 1960-as évek óta használja felderítésre és megfigyelésre, de ellenséges célpontok kiiktatására alkalmas felszereléssel is el tudják látni. Az előny elsősorban, hogy ezeket a repülőgépeket akár több 100 kilométeres távolságból is képesek irányítani, így veszélyes helyzetekben nem kell emberéleteket kockáztatni.

Napjainkban egyre többet olvashatunk arról, hogy pilóta nélküli repülőgépek permetezik a mezőket és a csomagok kiszállítását, vagy az elveszett síelők felkutatását is ilyen eszközök végzik. Ezeknek az objektumoknak a társadalmi és hétköznapi életben elterjedt elnevezése a drón. A drónok segítségével Kinában és Japánban automatikusan permetezik a rizsföldeket, [3] az Amazon nevű cég csomagokat szállít vele, [4] valamint Svájcban több síelő életét is megmentették. [5] Az utóbbi években elterjedt hobby alkalmazása is, viszont Magyarországon komoly légtérhasználati és egyéb szabályokhoz van kötve. Láthatjuk, hogy ezek az eszközök a világpiac több területén alkalmazhatók, legyen szó vészhelyzetről, vagy egy egyszerű videófelvétel készítéséről.

1.3. Motion Capture

A Motion Capture kifejezés magyar megfelelője a mozgásrögzítés. A mocap néven is elterjedt technológia lehetővé teszi bármilyen objektum mozgásának felvételét a háromdimenziós térben. Előszeretettel használják a filmiparban, ahol egy animált karakter mozgását egy valós színész végzi el. Elterjed az orvostudományban és a hadseregben is. Az objektumok dinamikus helyváltoztatása, a rájuk szerelt érzékelőkön, más néven markereken keresztül történik. Megkülönböztetünk aktív és passzív érzékelőket. Az aktív érzékelők közé tartoznak a LED, vagy egyéb más fényforrást használó megoldások. Ide sorolhatóak még a mágneses jelforrások is, viszont ezek nem annyira elterjedtek mint a LED érzékelők. A passzív érzékelők, az előzőekkel szemben csak fényvisszaverő tulajdonsággal bírnak. Ezek a markerek az infravörös fényt képesek visszaverni, melyek a felvételt végző kamerákra szerelt diódákból érkeznek. A diódák kimondottan erre a feladatra fejlesztett LED fényforrások. Fontos még megemlíteni, hogy csak a markerek háromdimenziós térben való mozgása kerül rögzítésre.

1.4. Neurális hálózatok

A neurális hálózat egy biológiai indíttatású architektúra, amit mesterséges neurális hálózatnak is szoktak nevezni, mivel az eljárás a biológiai neurális hálozat viselkedését modellezi. [6] Az elnevezés a neuron szóbol ered. A neuronok, más néven idegsejtek az idegrendszert alkotó legkisebb egységek. A mai értelemben vett neurális hálózat alatt, ezen ötlet alapján, programmal megvalósított architectúra értünk. A mesterséges neurális hálózat a legáltalánosabb formájában egy matematikai modell, ezt lehet programmal szimulálni, vagy akár hardverrel emulálni valamilyen programozható alkatrész segítségével.

Neurális hálózatnak nevezzük tehát azokat a hardveres vagy szoftveres megvalósítású információfeldolgozó eszközöket, melyek képesek párhuzamos, elosztott működésre. Hasonló vagy teljesen azonos, lokális feldolgozást végző, rendezett topológiájú neuronokból áll, amelyek nagymértékben összekapcsoltak egymással, és egységes rendszert képeznek. Rendelkeznek tanulási algoritmussal, mely általános esetben valamilyen minta alapján való tanulást jelent. Képes a megtanult információra hivatkozni, egy előhívási algoritmus segítségével.

1.5. Turing-teszt

A Turing-teszt egy Alan Turing nevű brit matematikushoz fűződik, akit a mesterséges intelligencia és a modern számítástudomány egyik atyjának is neveznek. A teszt azt hivatott eldönteni egy számítógépről, hogy képes-e olyan válaszokat produkálni mint egy hús-vér ember.

A teszt lefolyásának bemutatására több példa is létezik. Én, a számomra legérthetőbbet fogom ismertetni. A tesztben egy bíráló és két alany vesz részt. A bíráló és az alanyok közül az egyik ember, míg a másik mesterséges intelligencia. A bíráló egy számítógépen keresztül kommunikál a két alannyal külön-külön, és öt perce van megállapítani, hogy az alanyok közül melyik ember és melyik mesterséges intelligencia. Ezt a bíráló kérdések útján teszi. Ha az öt perces időintervallum letelte után nem képes megállapítani, hogy a két alany közül melyik melyik, akkor a mesterséges intelligencia átment a Turing-teszten. Fontos megemlítenem, hogy a kutatásomhoz nem kapcsolódik közvetlenül a Turing-teszt. Csak a későbbiekben való könnyebb megértést szolgálja ez a rövid leírás.

Egy másik hasonló eljárás a CAPTCHA nevű automatikus teszt. A CAPTCHA szó jelentése *Completely Automated Public Turing test to tell Computers and Humans Apart*, azaz teljesen automatizált nyilvános Turing-teszt a számítógép és az ember megkülönböztetésére. Ezt egyébként fordított Turing-tesztnek is szokták nevezni, mivel itt a számítógép teszteli a felhasználót, hogy valós személy, vagy pedig számítógép. A CAPTCHA teszt a Turing-teszthez hasonlóan gépek és emberek megkülönböztetésére szolgál. Az előbbi ezt feladványokon keresztül vizsgálja melyekkel a hétköznapi életben is sokszor találkozhatunk, ha például regisztrálunk egy honlapra, e-mail fiókot hozunk létre, vagy a bankfiókunkba szeretnénk belépni.

1.6. QR-kód

A Quick Response code, melynek magyar elnevezése a gyors válasz kód, a nevéből is adódóan gyors visszafejtési lehetőséget biztosít. [7] A QR-kód tulajdonképpen egy kétdimenziós vonalkód, melyet 1994-ben egy japán cég fejlesztett ki. A technológia mára igen népszerűvé vált, és az emberek hétköznapjaiban is fontos szerepet játszik.

Az ilyen kétdimenziós vonalkódok nagy előnye, hogy bármilyen irányból felismerhetőek, azaz nem szükséges a kód értelmezésénél a helyes tájolásra figyelni. Ezt a kódbélyegek sarkában elhelyezett azonos minták biztosítják, melyek alapján a kód szinte bármilyen szögből nézve értelmezhetővé válik. A QR-kód három sarkában lévő adatok alapján, ugyanis a dekódoló programok képesek eldönteni, hogy milyen irányban kell a kódbélyeg pontjait értelmezni.

1.1. ábra. QR-kód az adatok hozzáadása előtt. [8]

1.2. ábra. QR-kód az adatok hozzáadása után. [8]

1.3. ábra. Teljes QR-kód a mintákkal a sarkaiban. [8]

1.7. Felhasznált eszközök

Az egyetem UAV laboratóriumában több olyan hardveres és szoftveres eszköz van, melyek segítették a munkámat. Fontos megemlítenem, hogy ebben az alfejezetben csak általánosságban, nagy vonalakban mutatom be ezeket az eszközöket. A későbbi fejezetekben részletezem az eszközök jellemzőit.

1. Az első és legfontosabb objektum a drón, melyek közül több is a rendelkezésemre állt. A kutatásom emiatt univerzálisabbá vált, mivel a különböző repülőgépek eltérő paraméterei miatt egészen más eredményeket kaptam.

2. A drón helyzetének pontos meghatározásához szükségem volt a Motion Capture technológiára, melyet az OptiTrack cég által forgalmazott kamerarendszerrel és a hozzá tartozó eszközökkel valósítottam meg. [9]

3. A drón és a kamerarendszer adatainak kezelését, egy kimondottan erre a célra fejlesztett nyílt forráskódú rendszer szolgáltatta. A Robot Operating System, amely nagyszámú könyvtárat tartalmaz a robotikai alkalmazások fejlesztésének megkönnyítéséhez. [10]

1.8. Feladat ismertetése

A feladat a karon található UAV labor négyrotoros és X8-as elrendezésű repülőgépeinek beüzemelése, és egy kísérleti környezet megteremtése, majd pedig alapvető navigációs feladatok végrehajtása, melyeket a repülő csak a saját szenzorai és a kamerájából nyert információi segítségével végez el. Az általam választott feladat két, különböző eljárásokat használó, navigációs rendszer megvalósítása. Ezek után egy akadálypálya építése, majd a pálya segítségével a drónok autonóm átrepülése kamera és IMU adatok alapján. Az akadálypályán való navigációt képi adatok fogják elősegíteni. Egyszerű szöveges adat vagy akár bonyolultabb, zajjal szennyezett karaktersorozat értelmezés, és QR-kód detekció felhasználásával. A repülést az OptiTrack cég által gyártott és forgalmazott kamerarendszer segítségével fogom nyomon követni, és különböző méréseket végezni. A cél tehát egy-egy tesztrepülés, ahol szemléltetni tudom a repülőgépek szenzoraik és kamerájuk alapján történő navigációt.

2. fejezet

Előzmények

Az ebben a témában található szakirodalom jórésze megköveteli az adott projekthez tartozó eszközök beható ismeretét és gyakorlatban való alkalmazását. Abból kifolyólag, hogy a tudomány ezen területe igen fejlődőképes, a hasonló alkotásoknak, csupán elméleti háttere található meg a világhálon. A komoly pénzügyi háttérrel rendelkező cégek, kutatásaiknak forrásait majdhogynem hét lakat alatt őrzik. Ez persze természetes, viszont az általam elérhető, nyílt forráskódú projektek, gyakran hibásak, nem karbantartottak és alkalmatlanok az újrafelhasználhatóságra.

Ezek alapján próbáltam olyan munkákra hagyatkozni, amiket már többen pozitívan véleményeztek, és olyan kutatásokat tanulmányozni, melyek csak részben kapcsolódnak az én témámhoz, de megbízhatóak és jó kiindulópontot jelenthetnek.

2.1. Korábbi kutatások

Korábban már foglalkoztam a robotikával és a pilóta nélküli repülőgépekkel az önálló laboratórium című tárgy keretein belül. Ezalatt az idő alatt megismerkedtem az egyetem UAV laboratóriumával, ahol több drón jellemzőit is tanulmányoztam és közülük számos vezérlését is kipróbáltam. A számomra fontos paraméterek közé tartozott a megbízható, jó minőségű képet biztosító kamerarendszer és a kifejezetten beltéri használatra optimalizált felépítés.

A szakmai gyakorlatom alatt dolgoztam az OptiTrack kamerarendszerrel és elsajátítottam a hozzá tartozó Motive nevű szoftver alapszintű ismeretét. Megtanultam a kamerák megfelelő elhelyezését és a kalibrálás folyamatát, valamint a paraméterek helyes beállítását. Ez idő alatt több drón helyes vezérlését megtanultam. A kutatásom fő témája

a különböző repülőgépek és a kamerarendszer szoftvere közötti kapcsolat felállítása volt. Emellett a Motion Capture nevű technológia gyakorlati használatában is szereztem némi tapasztalatot.

Az előző szemeszterekben foglalkoztam a mesterséges intelligencia és neurális hálozatok témakörével. Nagy örömömre szolgált, hogy az ITK lehetőséget adott ezen technológiák megismerésére. Mint ahogy az én kutatásomban is, széles körben alkalmazható és sok, különböző tudományos területen felmerülő problémára megoldást nyújt.

A szakdolgozatomhoz kapcsolódó alkalmazási területe az optikai karakterfelismerés, amit az angol nyelvben OCR (Optical Character Recognition) elnevezéssel illetnek. [11] Az eljárás a kézzel írt, vagy gépelt dokumentumon található karaktereket képes beazonosítani, a dokumentum fotója alapján. A módszer részben hasonlít az objektum detekcióhoz, azzal a különbséggel, hogy itt az egyes elemek betűk, számok, vagy egyéb karakterek, nem pedig háromdimenziós objektumok. Egy korábbi projektemben CAPTCHA megoldó algoritmus elméleti hátterével és készítésével foglalkoztam. A CAPTCHA kifejezésről már írtam a bevezető során, most pár mondatban összefoglalom az optikai karakterfelismerés szemszögéből is.

Ebben az esetben egy adott képen található szövegről kell megállapítani hogy milyen karakterek szerepelnek rajta. Gyakran ezek a szövegek valamilyen zajjal szennyezettek. Itt a zaj a karakterek olvashatatlanságát előidéző vonalakat, színes pontokat, vagy akár áthúzásokat jelent. Találkozhatunk akár olyan fotókkal is, melyeken az egyes betűk részben egymáson, vagy valamilyen átfedésben helyezkednek el. Ez tovább nehezíti a feladatot.

A probléma megoldására több módszertan is létezik. Az egyik, hogy az adott képet többször tisztítjuk meg az esetleges zajoktól és felesleges részltektől, például vonalaktól, áthúzásoktól. A másik megoldási módszer megvalósításához valamilyen mesterséges intelligenciát használó eljárás szükséges. Ezeket a harmadik fejezetben fogom bemutatni.

2.2. Szakirodalom tanulmányozása

Fontos szempont volt a szakirodalmak, valamint szakcikkek tanulmányozásánál az adott irodalom, vagy cikk aktualitása. Próbáltam olyan publikációkat és kutatásokat választani, melyek ugyan nem az elmúlt évekből valóak, viszont olyan technológiát használnak, ami napjainkban is aktuális. Így kiszűrtem azokat az alkotásokat, amelyek nem megbízhatóak vagy pedig elavult módszereket használnak.

Az Egyesült Államokban található Indianai Egyetem egyik publikációja a pilóta nélküli légi járművek valós idejű objektum detektálását tárgyalja, melyet konvolúciós felhő alapú neurális hálózatokkal valósít meg. [12] A korábbi tanulmányaimból és az adott cikkből is megtudtam, hogy a konvolúciós neurális hálózatokra jellemzően a rendszer csúcskategóriás grafikus feldolgozó egységeket igényel. Ezek energiaszükséglete és súlya is meghaladja egy átlagos drón korlátait. A kutatás lényege, hogy a számításokat áthelyezik egy off-board számítási felhőbe, így mentesítve a repűlőgépet a terhelésektől. Ezzel a megoldással a drón több száz objektumot képes detektálni közel valós időben.

Jó példája volt ez számomra a mély tanulási eljárások alkalmazásának a robotika világában. Egy kis érdekesség, hogy a kutatás során a repülőgép alatti területet szerették volna elemezni, viszont a drón alsó kamerája nem adott elég jó minőségű képet, ezért kutatók az első kamera elé egy apró tükröt illesztettek, így irányítva a drón alatti területre a látószöget. Ez a kamera már megfelelő minőségű képet biztosított az objektumok detektálásához.

A szakcikkek tanulmányozása mellett a kutatás során felhasznált eszközök dokumentációjának megismerése is fontos volt. Elsajátítottam a kamerarendszer és a repülőgépek helyes használatát, továbbá számos felhasznált könyvtár és csomag használati útmutatóját is elemeztem.

2.3. Hasonló alkotások

Ebben a témában nagyszámú egyéni projekt található meg a világhálón. Sajnálatos módon ezek közül elég kevés az, ami hibamentes, jó dokumentációval rendelkezik, és alkalmas a részben vagy egészben való újrafelhasználhatóságra. Szerencsére sikerült találnom néhány kutatást, melyek hasonló módszereket használnak egy autonóm tesztrepüléshez, valamint az egyetem UAV laboratóriumában is készültek már ilyen jellegű alkotások. [13]

Az Amsterdami Egyetemen található ISLA (Intelligent Systems Laboratory Amsterdam) nevezetű labor keretein belül számos érdekes kutatást folytattak az utóbbi években. Többek között olyan követési algoritmusok autonóm kiértékelésével foglalkoznak, melyek dinamikus célpontokat vizsgálnak. Számomra talán a legaktuálisabb a látás alapú követéssel kapcsolatos munkájuk. [14] A projekt lényege egy autonóm átrepülés volt bizonyos akadályok között, majd pedig egy dinamikus és egy statikus célpont felismerése és esetleges követése. A célpontokat QR kódok képezték, melyek parancsokat tartalmaztak a drón számára. A kutatás során előállt problémák rémisztően hasonlóak, mint amiket én is tapasztaltam munkám során. Egy kritikus pont volt a QR kód felismerése, mivel a drón állandó légi mozgást végez, ami a kamera képének elmosódásához vezet. Ennek megoldására a Zxing nevű könyvtárat alkalmazták, mely jó felismerési algoritmust biztosított ilyen helyzetekben is. Abban az esetben, amikor a repülőgépnek követni kellett a megadott célpontot megbízható nyomon követési eljárásra volt szükség. Az OpenTLD egy nyílt forráskodú C++ implementációja a TLD (Tracking Learning Detection) eljárásoknak, ami hasonló feladatokat hivatott megoldani. [15] Többek között itt is ezt a módszert használták a probléma megoldására.

Sok új ötlettel és lehetőséggel gazdagodtam a korábban említett projekt elemzése során, továbbá azt is megtapasztaltam, hogy az elméletben jól alkalmazható modell a gyakorlatban nem mindig praktikus megoldás.

3. fejezet

Tervezés

A szakirodalom és a felhasznált eszközök dokumentációjának tanulmányozása után következhetett a tervezés folyamata. Ebben a fejezetben részletesen bemutatom a fejlesztés során szerzett tapasztalataimat, a kutatás során használt módszereket és a lehetséges megoldásokat egy kamera alapú beltéri navigációs rendszer megvalósításához.

3.1. A kutatás során felhasznált objektumok

A munkám elvégzéséhez számos hardveres és szoftveres eszköz segítséget nyújtott. Ebben az alfejezetben ezeket az objektumokat szeretném bemutatni.

3.1.1. Repülőgépek

A munkámhoz használt repülőgépek kiválasztása során több szempontot is figyelembe vettem. Az elsődleges szempont, hogy megbízható kamerarendszerrel legyen felszerelve és kimondottan beltéri navigációhoz legyen kalibrálva. Ilyen paraméterekkel két drón rendelkezett, az egyetem UAV laboratóriumából.

Az egyik a Parrot cég által gyártott AR. Drone 2.0 repülőgép volt, mely a 3.1. ábrán látható. Az eszköz egy távolról vezérelhető és kifejezetten beltéri használatra alkalmas négyrotoros pilóta nélküli repülogép. [16] A kommunikációt más eszközökkel Wi-Fi kapcsolaton keresztül képes megvalósítani. Fontos megemlíteni a 4 GB-os memóriát is, mely lehetővé teszi az egyedi algoritmusok tárolását, és felhasználását. A tárhely segítséget nyújt a feladatok automatizálásában, a szenzorok és a kamerák által nyert adatok kiértékelésében is. Két kamerával van felszerelve, melyek a drón első és alsó részén helyezkednek el. Az első kamera jó minőségű képet biztosít a felhasználójának. A drón alsó részén található egy ultrahangos távolságmérő és egy légnyomásmérő szenzor, továbbá be van építve

egy 3 tengelyű giroszkóp, egy 3 tengelyű gyorsulásmérő szenzor és egy magnetométer, amik a drón stabilan tartását hivatottak biztosítani. Ezek az érzékelők teremtik meg a beltéri repüléshez, és kísérletezéshez szükséges adatokat amiket a drón a navigációhoz is felhasznál. A repülőgép méretét tekintve, az 57 centiméteres hossza és ugyanekkora szélessége igen nagynak mondható a beltéri használathoz.

3.1. ábra. A Parrott cég AR. Drone 2.0 elnevezésű repülőgépe. [16]

A másik a Ryze Tello Edu elnevezésű drón volt, mely DJI technológiával készült. [17] A mindössze 17,8 centiméter hosszú és 16,8 centiméter szélességű repülő sok pozitív tulajdonsággal rendelkezik a beltéri alkalmazások terén. Méretéből adódóan kiválóan navigálható, valamint 5 megapixeles kamerája elektronikus képstabilizátorral van ellátva, így alkalmas jó minőségű képek biztosítására. Az on-board vezérlést az Intel cég által gyártott 14 magos processzor hajtja végre. A drón alsó részén található egy lézeres távolságmérő és egy légnyomásmérő szenzor, továbbá be van építve egy 3 tengelyű giroszkóp, egy 3 tengelyű gyorsulásmérő szenzor és egy magnetométer, amik a drón stabilan tartását hivatottak biztosítani. Nagy hátránya, hogy az általam használt Robot Operating System nem tartalmaz hozzá annyi könyvtárat és hasznos modult, mint versenytársaihoz. Ez persze abból is adódik, hogy a A.3. ábrán látható repülőgép, alig csak pár éve került forgalomba.

3.2. ábra. A DJI és a Ryze cégek Tello Edu elnevezésű repülőgépe. [18]

3.1.2. Robot Operating System

A Robot Operating System egy nyílt forráskódú keretrendszer, melyet meta-operációs rendszernek is neveznek, mivel biztosítja a hardver szintű absztrakciót, továbbá képes kezelni a különböző hardverek vezérlését és az egyes folyamatok közötti kommunikációt.

A ROS néven is ismert rendszer, számos olyan könyvtárat és egyéb eszközt tartalmaz, mely nagyban megkönnyíti a robotikával kapcsolatos fejlesztéseket. [19] A nyílt forráskódból adódóan elősegíti a kód újra felhasználást, amelyek közül rengeteg elérhető a hivatalos weboldalán. Fontos megemlítenem még a hozzá tartozó rviz nevű vizuális megjelenítőt, ami az esetleges kamera kép megjelenítése mellett, más fontos információkat is közöl a felhasználóval. Képes nyomon követni a repülőgépek és robotok navigációs adatait és egyéb paramétereit, valamint egy szimulációkhoz használható háromdimenziós tér megjelenítő is elérhető benne. A keretrendszer C++ és Python programozási nyelveket támogat.

Mivel a teszteléseket két különböző repülőgépen végeztem, a gépek eltérő paraméterei és kompatibilitása miatt, más ROS verziót, és ebből kifolyólag más Linux operációs rendszert kellett használnom. Ez abból is adódik, hogy a AR. Drone 2.0 jóval régebben forgalomban lévő eszköz, ezáltal régebbi rendszereket is támogat, míg a Ryze Tello Edu elnevezésű repülő újabb verziókkal működik hibamentesen. A drónokhoz tartozó könyvtárak és a fejlesztések során segítséget nyújtó eszközök is a Robot Operating System

eltérő disztribúciót preferálják. Ez a probléma, nagyban megnehezítette a munkámat és sok időt igénybe vett a különböző rendszerek konfigurálása.

Az AR. Drone 2.0 repülőgéphez egy valamivel korábbi ROS verzióra volt szükségem. Szám szerint a tizedikre, melyet a Kinetic névvel illetnek. Ehhez a Lubuntu 16.04 LTS Xenial verzióját használtam, ami szintén egy, már évek óta használatban lévő rendszer. Mint utólag kiderült, ez nem volt túlságosan jó választás, mivel a Lubuntu rendszer nem támogatta a QR-kód detektálására alkalmas *ar_track_alvar* nevű ROS csomagot.

A Ryze Tello Edu repülőgéphez a legújabb ROS disztribúciót alkalmaztam, amely csak azt a fontos újítást tartalmazta számomra, hogy az előzővel szemben, ezt a verziót támogatja a Tello. Ehhez az Ubuntu 18.04 LTS Xenial verzióját használtam.

A Robot Operating System egy igen sajátos szemantikával rendelkező operációs rendszer az általam eddig ismertekkel szemben. A futtatható állományok üzeneteken keresztül kommunikálnak, feliratkozások és publikációk segítségével, melyeket témákba szervezve lehet elérni. Egy rövid felsoroláson keresztül megpróbálom bemutatni a rendszer építőelemeit. [20]

1. Node: Olyan futtatható állomány, amely a ROS rendszeren keresztül kommunikál másik futtatható állományokkal.

2. Message: A node kommunikációs primitív adattípusai, vagyis ROS adattípusok, amelyeket topic feliratkozásra vagy publikációra használhatunk.

3. Topic: A node egységek üzeneteket publikálhatnak egy adott témába, azaz topic-ba, valamint feliratkozhatnak témákra, így megkapják az oda publikált üzeneteket.

4. Service: Egy adott futtatható állomány szolgáltatása, mely más node számára igénybe vehető.

5. Master: A Robot Operating System központi egysége, amely a node és topic elemekről tárol információkat, továbbá segíti az elemek megtalálását.

6. Parameter Server: Központi tároló egység, mely a node elemek paramétereit tárolja.

7. Rosout: Ez az egység gyűjti össze a node elemek debug kimenetét. A C és C++ nyelvekben használatos stdout-hoz hasonlítható, azaz a szabványos kimenetet képezi a különböző egységeknél.

8. Roscore: A ROS központi magja, amely a Rosout, Parameter Server, és a Master együtteséből épül fel. A rendszer ezen eleme minden esetben fut.

3.1.3. OptiTrack kamerarendszer

Kamerák

A bevezető során már említést tettem a Motion Capture technológiáról és annak gyakorlati alkalmazásáról. A hardver, amely képes biztosítani számomra ezt, az OptiTrack cég által forgalmazott kamerarendszer. A VICON cég mellett a másik piacvezetője az ejárásnak. [21] Ezek a speciális eszközök, az infravörös fényt megfelelő módon átengedő lencsékkel és szűrőkkel vannak felszerelve, valamint a lencséjük körül megtalálható egy gyűrű, amely infravörös fény kibocsátására alkalmas diódákat tartalmaz. Egy másik fényforrással is ellátták a kamerákat, melyek egy adott beazonosítását, és a kalibrálás folyamatát segítik elő. Ezek az úgynevezett állapotjelző diódák szintén gyűrű alakban helyezkednek el a lencse körül.

Az egyetem UAV laboratóriuma az OptiTrack Prime 13 típusú kamerával rendelkezik. Ebből 8 darab állt a rendelkezésemre, ami bőven elég volt egy ilyen méretű teszt elvégzésre. A kamerák 13 megapixeles felbontóképességgel rendelkeznek, 1280 x 1024 pixeles képarány mellett. Képesek akár 240 képkockát rögzíteni másodpercenként. Ezt vertikálisan 46, horizontálisan pedig 56 fokos betekintési szög mellett teszik lehetővé. Az átlagos késleltetési idejük 4,2 ms, ami kimondottan alkalmas a valós idejű mozgáskövetéshez átlagos helyzetek esetén. Persze, ha egy vadászrepűlő mozgását szeretnénk megfigyelni, ez nem megfelelő beállítás, de az én kutatásomat tökéletesen kiszolgálja. A viszonylag kis betekintési szög miatt, és a tesztkörnyezet méretéből adódóan a mozgáskövetés 8 kamera használata esetén ad pontos eredményt. A kamerák ethernet kábelen keresztül kommunikálnak az őket vezérlő számítógéppel és ugyanezen a kábelen tudják felvenni a működésükhöz szükséges áramot. A switch kiválasztásánál ez fontos szempont volt, mivel olyan eszközre volt szükségem, amely képes a PoE, azaz Power over Ethernet funkcióra. [22] A tervezés fejezetében ezt az eszközt is bemutatom. A kamera a 3.3. ábrán látható.

3.3. ábra. Az OptiTrack cég Prime 13 típusú kamerája technikai specifikációval. [23]

Markerek

A markerekről is tettem már említést korábban. Ebben a részben átfogóbban jellemzem az általam használt passzív típusú markereket.

Ezek olyan gömb alakú objektumok, melyek felülete nagy hatékonysággal képes visszaverni a felületére érkező fényt. Ezt én is többször megtapasztaltam, amikor a nap fénysugarai érték a markereket. A különböző hullámhosszú fények közül a magas, azaz az infravörös fényt képesek a leghatékonyabban reflektálni. Ez fontos tulajdonság különösen az előbb említett jelenség miatt, miszerint a nap sugarait is visszaverik, viszont a kamerák ezt teljesen képesek elkülöníteni az általuk kibocsájtott magas hullámhosszú infravörös fénytől. A markereket gömb alakjuk miatt a kamerák pontforrásként érzékelik.

3.4. ábra. Egy rögzítő talppal ellátott fényvisszaverő marker. [24]

Fontos megemlítenem a markerek azon csoportját, melyek a tesztrepülés során a drónra szerelve kapnak helyet. Ezen markercsoport pozíciójának és alakjának segítségével képes a rendszer a repülőgép adatainak nyomon követésére. A markereket egy speciális elrendezésű testen kell elhelyezni, így egy egységet képeznek. A test megközelítőleg 18 centiméter hosszú és 16 centiméter széles. Az speciális elrendezésből kifolyólag az OptiTrack szoftvere képes a repülőgép pozíció, orientáció és gyorsulás adatainak meghatározására. A test a 3.5. ábrán látható.

3.5. ábra. A rigid body elnevezésű speciális test, a markerek felszerelése előtti állapotban. [25]

Motive

Az OptiTrack cég termékeihez tartozó hivatalos szoftver a Motive. A kamerákból származó adatok értelmezését és a háromdimenziós tér megjelenítését végzi, valamint nagyszámú konfigurációs lehetőséget biztosít a felhasználónak. A kamerák UDP kábeleken keresztül kapcsolódnak közvetetten a programot futtató számítógéphez. A kamerákból érkező vezetékeket először egy switch-hez kell közvetlenül csatlakoztatni, amihez a számítógép is közvetlenül kapcsolódik. A program elindítását követően, egyből láthatóvá válnak a kamerák képei.

A rendszernek szüksége van a kamerák kalibrációjára háromdimenzió tér érzékeléséhez. Az eljárás neve *wanding*, ami nagyjából annyit takar, hogy a kamerák és egy kimondottan erre a célra kifejlesztett eszköz segítségével több ezer mintát gyűjtünk. A 3.6. ábrán látható kalibráló pálca néhány percen át tartó mozgatása szükséges a kamerák által megfigyelt területen. Ezt követően a szoftver kiszámítja a kamerák egymáshoz viszonyított pontos pozícióját és legenerálja az általa érzékelt háromdimenziós teret. Szükséges még a

talajszint meghatározása, melyet a kalibráció befejezte után, a *ground plane* nevű eljárással végezhetünk. Ehhez is egy speciális eszközre van szükség, mely a 3.7. ábrán látható.

3.6. ábra. A talajszint meghatározására alkalmas CS-200 jelölésű kalibráló elem. [26]

3.7. ábra. A kalibrálás során használt CW-500 jelölésű pálca a hozzá tartozó markerekkel. [26]

A program lehetősget biztosít különböző koordináta transzformációk elvégzésére, valamit markerek egy csoportját összevonhatjuk és egy objektumként definiálhatóvá válnak. A munkám során két marker csoportot hoztam létre, melyek közül az egyik a drón helyzetét jelezte, a másik pedig a célállomás volt.

3.1.4. Számítógépek

A tesztelés során 2 számítógépre volt szükség. A Motive szoftver, amely Windows operációs rendszer felett futott és közvetlenül csatlakoztatva volt egy switch-hez, amihez a kamerarendszer. A másik számítógépen a Robot Operating System futott, néhány kiegészítő csomaggal egyetemben. Ezt az eszközt is csatlakoztattam a switch-hez, így a Motive adatai elérhetővé váltak. A rendszer vezérlése ezen a számítógépen történt, mivel ide érkeztek be a kamerarendszer és a drón adatai is, valamint erről a platformról volt lehetőség utasításokat adni.

3.1.5. Switch

Már korábban említést tettem a PoE, azaz a Power over Ethernet funkcióról az OptiTrack kamerák bemutatásánál. Az ilyen típusú kamerák nem rendelkeznek külön áramellátással. A működésükhöz szükséges áramforrás ethernet adathálózaton keresztül van biztosítva, melyet a kommunikáció során is használnak. A specifikáció maximum 48 V

feszültséghez 350 mA áramot enged meg. Ez bőségesen elég egy Prime 13 típusú kamera tápellátásához.

A kamerák számára az áramot egy kimondottan erre kifejlesztett switch szolgáltatta, amely képes a fentebb említett funkcióra. A munkám során a NETGEAR cég ProSafe GS728TPP típusú switch állt a rendelkezésemre, mely a 3.8. ábrán látható. Az eszköz 24 darab gigabit ethernet prottal van felszerelve és összesen 384 W teljesítményre képes. Az üzembe helyezés semmilyen előzetes konfigurációt nem igényelt, az esetemben 8 darab ilyen portot használó rendszer felállításához.

3.8. ábra. A tesztelés során használt NETGEAR ProSafe GS728TPP típusú switch. [4]

3.1.6. Router

A teszt elvégzéséhez szükségem volt, még egy eszközre. A hálózati beállítások pontos paraméterezésére és az esetleges kapcsolati hibák orvosolására egy Routert használtam. Erre azért volt szükség, mivel az általam használt ROS csomagok közül néhány eszköz a Wi-Fi hálózaton keresztüli kommunikációt támogatta. Erről a hálózati beállítások alfejezetben részletesen beszámolok.

Az általam használt LINKSYS WRT1900AC típusú router tökéletesen kiszolgálta a feladatait a kutatás során. Képes az 5 GHz tartományban való kommunikációra, és ezáltal a legújabb AC szabványok használatára. Egy kritikus pont volt, hogy repülőgépek fedélzeti számítógépe kizárólag olyan hálózatokhoz tud csatlakozni, melyek nincsenek titkosítva. A router képes erre a funkcióra előzetes beállítások nélkül, valamint a Motive adatait is tudtam ezen a csatornán keresztül venni, a ROS rendszert futtató számítógéppel, ha a switch által szolgáltatott kapcsolat nem működött hibamentesen. Fontos megjegyez-

nem, hogy ez nem a switch konfigurációjából, hanem a hálózatért felelős ROS csomagok elavultságából adódótt. A router a 3.9. ábrán látható.

3.9. ábra. A LINKSYS cég által forgalmazott WRT1900AC típusú router. [27]

3.2. Szövegfelismerő eljárás bemutatása

Az egyik ötletem az volt, hogy az repülőgép autonóm vezérlését szöveges információ alapján fogom működtetni. Az ehhez használandó technológiát nevezhetjük optikai karakterfelismerésnek is, melyet az előző fejezetben röviden bemutattam. Az első teszt tehát, egy szövegfelismeréssel vezérelt autonóm repülés egy általam épített akadálypályán keresztül.

A szövegfelismerő eljárás tesztelését az AR. Drone 2.0 eszközön végeztem. A repülőgép kamerája és további paraméterei, valamint kompatibilitási képessége jobb esélyekkel kecsegtettek.

A fejlesztés megkezdése előtt találkoztam az interneten egy hasonló projekttel, ami szintén szöveges adatok alapján oldotta meg a repülőgép autonóm vezérlését. [28] A kolozsvári Egyetem hallgatói egyszerű angol szavakkal vezérelték a drón mozgását. A left szó a balra, míg a right a jobbra fordulást idézte elő. Ugyanígy történt a leszállás hívása is. A kutatáshoz a LabVIEW nevű szoftvert használták, így számomra csak elméleti segítséget nyújtott, de kellemes meglepetés volt, hogy mások is hasonló feladatokon dolgoznak és sikerrel járnak.

3.2.1. Szövegfelismerés CAPTCHA megoldó algoritmussal

Az egyik lehetséges módszer egy CAPTCHA megoldó algoritmus logikáján alapuló eljárás implementálása, majd pedig integrációja a navigációs rendszerbe. [29] Érdekes kérdés, hogy egy ilyen rendszer esetén miért lehet előnyös, ha egy bonyolultabb és nehezebben kezelhető adat feldolgozására van optimalizálva.

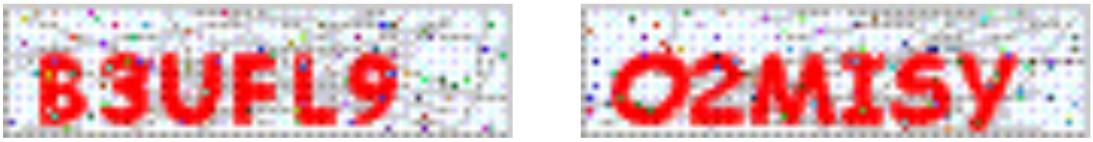
Az elképzelésem az volt, hogy sikeresen hasznosíthatom a korábbiakban ezen a területen szerzett tapasztalataim, valamint így a drón komplexebb feladatok elvégzésére is alkalmas lesz. Fontos szempont volt még, hogy ez a módszer a repülőgép manőverezéséből és instabilitásából adódó gyengébb minőségű képek elemzését is lehetővé teszi. Ugyan a korábbi kódomat több ponton is tovább kellett fejlesztenem és sok átalakítást igényelt, így is jó esélyekkel kecsegtetett. A nehézséget az jelentette, hogy mikor a drón a kameráján keresztül érzékelte a szöveget, egy fényképet készített és ebből a képből még ki kellett nyernem azt a bizonyos részt, melyen a drónnak szánt üzenet volt.

Megoldási módszerek elemzése

A képi adatok feldolgozására, majd az ebből való információ kinyerésére több módszer is a rendelkezésemre állt. Az egyik, hogy az adott képet többször tisztítom meg az esetleges zajoktól és felesleges részektől, például vonalaktól, áthúzásoktól és egyéb zavaró tényezőktől, majd a kapott képet szürkeárnyalatosra módosítom. Ezután ismét eltávolítom az így detektálhatóvá vált zajokat. A következő lépés, hogy szeparálom a karaktereket, ha lehetséges, mivel a bonyolultabb CAPTCHA feladványoknál találkozhatunk olyan esetekkel is, ahol erre nincs lehetőség, mert a karakterek szélei fedésben vannak egymással. A szeparálás a karakterek határainak keresésével kezdődik, ha pedig ezek fedésben vannak, akkor akár egyben is vizsgálhatóak. Erre léteznek speciális módszerek, melyek az átfedés mértékétől, valamint további zavaró tényezőktől is függnek. A kutatásom során ezeket csak elméleti szinten érintettem és az esetleges továbbfejlesztési lehetőségek alapjait szolgálta.

Miután megkaptam a szétdarabolt karakterekkel, ezeket össze kell hasonlítani egy már előre elkészített karaterhalmazzal. Az eljárás ezen része némileg hasonlít egy neurális hálozat működéséhez. A képdarabokon pixelenként végigfutni, és megvizsgálni a hasonlóságot a meglévő halmaz elemeire. A lehetséges találatokat tárolni kell és további vizsgálatokat végezni rajtuk. Ezt addig szükséges végezni, amíg valamilyen hibahatáron belülre kerülnek az adott karakterek.

A másik megoldási módszer megvalósítása, a már korábban említett mesterséges intelligencia segítségével történik. A eljárás lényege, hogy egy tanulóhalmazt hozunk létre, és ezen halmaz segítségével oldjuk meg az adott problémát. Sok ilyen algoritmus és feladatmegoldó metódus létezik. Egy ígéretes választás lehet a Multivalued Image Decomposition nevű eljárás, amely a kép színeinek vizsgálatával képes megoldást nyújtani a problémára. [30] Ehhez a módszerhez nagy segítséget nyújt a Python programozási nyelv egyik könyvtára. A PIL, azaz a Python Imaging Library nevű modul. Ennek segítségével a fentebb említett algoritmus kényelmesen implementálható.



3.10. ábra. Az eljáráshoz tartozó, különböző zajokkal szennyezett képek, melyek a repülő számára információt hordoznak. [29]

Megoldási módszer bemutatása

A végső megoldás a két módszer ötvözetéből állt elő. Az algoritmus megvalósítása során a Python 2.7.13 verzióját használtam. Erre azért volt szükség, mert a PIL (Python Imaging Library) 1.16-os verziója ezt támogatja. Továbbá felhasználtam a népszerű numpy package-t is.

Az aktuális képben lévő szöveg kinyerésére és felismerésére a már említett Multivalued Image Decomposition nevű algoritmust használtam. Ez egy olyan egyszerű eljárás, amely első lépésként a kép színeiből egy hisztogramot készít, ezért a kép minden egyes pixelét ellenőrizni kell. A pixelek színének meghatározása után a képen található minden színhez számolható egy érték, amely azt mutatja, hogy az adott szín hány pixelen jelenik meg.

A színek meghatározásához a megfejtendő képet GIF formátumban tároltam el, mivel ez a formátum csak 256 különböző színt tud kezelni, így a hisztogram készítése egyszerűbbé vált. A PIL modul használatával ezek a műveletek könnyen elvégezhetőek voltak.

Az eredmény úgy alakult, hogy a leggyakoribb szín a háttér, amelyet el kellett távolítani, ezt a karakterek színe követte, majd pedig a hibás pixelek, a szándékos torzítás és a zaj. Tehát a második leggyakoribb szín megtartásával, egy fekete-fehér képet kaptam. A többi szín eltávolításával nagy arányban kiszűrhetővé váltak a különböző zajok.

Az megoldás csak egyszínű szöveget képes jó aránnyal felismerni, de az algoritmus könnyen átalakítható úgy, hogy több színű betűk felismerésére képes legyen. Ehhez a képen található színek alapján több fekete-fehér képet kell készíteni, és ezeket a képeket összevonni. A későbbiekben a színes képek értelmezését is el szeretném érni.

A kép darabolása során azt kellett megvizsgálnom, hogy egy beállított, azaz fekete színű pixel szomszédos-e egy szintén már beállított pixelnek. Ha az előbbi állítás igaz, akkor a pixelek egy egységbe kerültek, ha viszont ez nem teljesült akkor egy új elemet kell kezdeni. A végeredmény pedig karakterekre bontott képek halmaza.

A korábbiakban említettem, hogy a megoldás során szükség volt valamilyen mesterséges intelligenciát alkalmazó eljáráshoz. A legkézenfekvőbb módszer a Vector Space Image Recognition volt számomra.

Adatok klasszifikációjára egy jó alternatíva a Vector Space Search Engines. Ennek több előnye is van, például nem szükséges nagyméretű tanító eljárás és túltanítani sem lehet. Természetesen nem tökéletes, hiszen a neurális hálókkal szemben jóval lassabban osztályoz.

Az előzőek alapján tehát szükséges egy tanítóhalmazt előállítani, illetve ezzel az AI tanítása. Minden felismerendő karakterhez legalább egy példa a program rendelkezésére állt. A karakterek képei egy gyűjteménybe szervezve voltak elérhetőek a futtatott kód mellett.

Sajnálatos módon a navigációs rendszerbe való integráció csak részben sikerült. A drón kamerájának képéről egy pillanatfelvételt kellett készíteni, amikor egy szöveges ablakot

érzékel. Ezután a felvételből csak az információt hordozó, azaz a papíron lévő karaktersorozatot volt szükséges beazonosítani, majd ezt körbevágni és tárolni. A tárolt képet GIF formátumra alakítani és a már fentebb említett eljárással értelmezni. Ezt követhette volna a konkrét utasítás elvégzése.

Az algoritmus GIF formátumú képekre szépen működik és a drón képének GIF formátumra való átalakítását is elvégzi. A fő probléma, hogy repülőgép kamerája a szöveges ablakot nem derékszögben látja a fénykép készítésének pillanatában, és a már korábban említett QR-kóddal szemben ez gondot jelent. Ez a karakterek értelmezését nagyban megnehezíti, mivel az egyes elemek különböző szögekből egészen más alakúak és gyakran egymásra is hasonlítanak. Jó példa erre a 0 és az O katakterek ahol, ha egy kicsit más szögből néztem az egyik elemre, máris kísérteties egyezést véltem felfedezni. A tájolás tehát nagy szerepet játszik ebben az esetben a karakterek értelmezésében.

Abban reménykedtem, hogy a CAPTCHA megoldási eljárás némileg segíteni fog a probléma megoldásában. Az biztosan kijelenthető, hogy a drón manőverezéséből és instabilitásából adódó zajokat hatékonyan kiszűri, viszont a karakterek más szögben való beazonosítását nem segíti. A tényleges megoldást a már korábban említett tanítóhalmaz bővítése jelentheti. A felismerendő karakterekhez több példát szükséges megadni, melyek más-más szögből látják az adott elemet, így sokkal biztosabb találati eredmény érhető el.

A felvétel elemzése után a konzolon megjelennek párok, amelyek első értéke azt jelzi, milyen valószínűséggel sikerült az adott karaktert felismerni, mellette szerepel a felismert karakter, mely az algoritmus alapján a legjobb hasonlóságot eredményezte.

Az említett algoritmus mellett, szükségem volt a Robot Operating System és az AR. Drone 2.0 repülőgép közötti kapcsolat felépítésére. Erre a ROS rendszerben több csomag is a rendelkezésemre állt, melyek nagy segítséget nyújtottak és sok egyszerűsítési lehetőséget biztosítottak. Az egyedüli hátránya ezeknek a forráskódoknak, hogy állandó fejlesztés alatt állnak és ebből kifolyólag kompatibilitási problémák léphetnek fel a régebbi disztribúciók estében.

Az egyik általam használt csomag az *ardrone_autonomy* volt, amely egy driver az általam használt AR. Drone 2.0 repülőgéphez. Felelős a drón navigációs adatainak, valamint kamera képének biztosításáért, és más fontos paraméterek beállítását is elvégzi.

Egy másik elem fölé épült, amely egy hardverközeli eszköz, és közvetlenül a drón fedélzeti számítógépével kommunikál. A modul apróbb hibák mellett helyesen működött a fejlesztés során.

A másik csomag az *ardrone_ tutorials* volt. Számos hasznos funkcióval van felszerelve, melyek például a drón irányítására használatos eszközökhöz tartozó vezérlés implementációja. Ehhez tartozik egy bemutató is, amely a vezérklők kezelését segít elsajátítani. [31] Ami számomra fontos volt, hogy ez a csomag az *ardrone_ autonomy* fölé épült, így további egyszerűsítéseket nyújt az autonóm vezérlés megvalósítására, valamint az implementációra vonatkozóan.

3.3. QR-kód felismerő eljárás bemutatása

A másik ötletem egy QR-kód felismerő eljárás segítségével való vezérlés volt. Ennek a megvalósítása jóval egyszerűbb a szövegfelismerésnél és több általam is elérhető irodalom található a témában. A második teszt tehát, egy QR-kód felismeréssel vezérelt autonóm repülés egy általam épített akadálypályán keresztül, ahogy az előző esetben is.

Fontosnak tartottam, hogy a fejlesztés megkezdése előtt megismerkedjek olyan kutatásokkal melyek az eljárás gyakorlati alkalmazását tárgyalják. Rövid keresés után rátaláltam az Aeriu elnevezésű magyar startup cégre. [32] A szolgáltatásaik a logisztikai automatizálás területére vonatkozik. A cég hatalmas raktárakban történő leltározást végez, ahol a repülőgépek autonóm módon képesek végighaladni a raktár egyes sorain, miközben képesek a két- és háromdimenziós vonalkódok leolvasására. A projekt kellemes inspiráció számomra, mivel ahogy én is beltéri navigációs rendszer fejlesztését tűztem ki célul, úgy az Aeriu is beltéren aktuális. Egy, a cégről szóló cikkben, a fejlesztők ki is emelték, hogy mivel GPS alapú navigációra nincsen lehetőségük, egészen új és egyedi technikákat kellett alkalmazniuk, ami persze nem publikus. [33]

3.3.1. QR-kód felismerés a ZBar ROS csomaggal

A szövegfelismerő eljárással szemben, itt az információkat különböző QR-kódok szolgáltatták a repülő számára. Ez a megoldás több szempontból is egyszerűbben kivitelezhető. A szövegfelsímerésnél fennálló probléma, miszerint a repülő kamerájának a karaktersorozatot, pontosan szemből kellett látnia, ebben az esetben nem volt szükséges. Ahogy már a bevezetőben is említettem, a QR-kódok esetén a kódbélyeg három sarkában található

jellegzetes minta alapján a kód szinte bármilyen szögből értelmezhető a program számára. A másik előnye ennek a technológiának, hogy gyengébb minőségű képet biztosító kamerák esetén is elvégzi a feladatot.

Megoldási módszerek elemzése

A Robot Operating System több kétdimenziós vonalkód olvasására alkalmas csomagot tartalmaz. Ezek közül számomra csak azok feleltek meg, melyek ilyen instabil esetben is képesek ellátni a feladatukat.

Az egyik lehetőség az *ar_track_alvar* elnevezésű csomag volt, mely a témában található források alapján a legalkalmasabbnak tűnt. [34] Az általam tanulmányozott korábbi kutatások is rend szerint ezt a modult használták a feladat elvégzéséhez, mivel kimondottan instabil képi adat feldolgozására lett optimalizálva és széleskörű támogatottsággal rendelkezik.

A másik lehetsőség a *visp_auto_tracker* nevezetű multifunkciónális csomag volt. Képes kamera által észlelt QR-kód követésére, valamint több kódbélyeg egyidejű beazonosítására is. A modul a Visual Servoing Platform egyik eleme, amely más hasznos csomagokat is tartalmaz, mint például háromdimenziós objektumok érzékelés és követése, továbbá a *visp_camera_calibration* nevű csomag egyedi mintákhoz való kamera kalibrációk elvégzését teszi lehetővé. [35] A jelen kutatásom elvégzéséhez ezek a funkciók nem szükségesek, valamint nagy hátránya, hogy a csomagok magas minőségű, stabil képek esetén működnek hibamentesen. A világhálón megtalálható cikkek alapján is arra a következtetésre jutattam, hogy inkább földön történő robotikai alkalmazásokhoz előnyös választás.

A harmadik és egyben utolsó lehetőség a *zbar_ros* nevű csomag volt. A témában talán a legelterjedtebb és legáltalánosabb nyílt forráskódú szoftvercsomag. Számos nyelvhez megtalálható az implementációja és széles körben támogatott. Az egyik legnagyobb előnye, hogy kifejezetten gyenge képminőség esetén is képes a QR-kód detekcióra. Nincsen felszerelve semmilyen további funkcióval, a *visp_auto_tracker* modullal szemben, de a jelenlegi kutatásomhoz, ez a csomag rendelkezik a legjobb paraméterekkel. Néhány vélemény szerint az instabil képek feldolgozása problémát jelenthet az algoritmus számára, de ezt a gyakorlatban szerencsére nem tapasztaltam.

Megoldási módszer bemutatása

A korábban bemutatott lehetőségek közül, ahogy a fejezet címéből is következik a *zbar_ros* csomagot használtam. A döntésemet nagyban befolyásolták egyéb más modulok melyek használata szükséges volt a repülőgép és a Robot Operating System helyes kommunikációjához, valamint a képi adatok kezeléséhez.

Az egyik fontos egység a *tello_driver* nevű csomag volt, amely tulajdonképpen az egyetlen interfész a ROS rendszer és a Ryze Tello Edu típusú drónok között. Ez a modul a TelloPy elnevezésű Python nyelvű csomag fölé épült, amely alacsony szintű, harverközeli elem, és ebből kifolyólag jó alapjául szolgált a *tello_driver* csomagnak, mely néhány kompatibilitási probléma mellett megbizhatóan működött.

Fontos megemlítenem, hogy a ROS csomagjai a különböző disztribúcióktól függően eltérő forráskódúak lehetnek és állandó fejlesztés alatt állnak, ami sokszor problémákat okozhat. Nem volt ez másként az én projektem során sem. Ezeket az esetleges problémákat gondos tervezéssel és megbízható elemek használatával próbáltam kivédeni, ami részben sikerült is.

Pontosan az ilyen problémákból fakadóan, szükségem volt egy másik csomag használatára. Az *image_transport* nevű modult, minden esetben alkalmazni kell a képi adatok kommunikációja során. Az én esetemben viszont ez egy kissé bonyolultabb volt. A csomag alapesetben kis sávszélességű *compressed* típusú információt közöl, amit viszont a *tello_driver* csomag nem képes fogadni, így szükséges volt az adatokat *raw* típussá alakítása. A megfelelő típus kinyerése után a program ezen része hiánytalanul működött.

3.4. Hálózati beállítások

A hálózati beállítások a két teszt esetén eltérőek voltak a különböző ROS disztribúciókból adódóan, valamint a repülőgépek méretére vonatkozó paraméterek miatt. Ebben a alfejezetben a hálózatok konfigurálására vonatkozó lépéseket, és az ehhez szükséges ROS csomagokat fogom bemutatni.

3.4.1. Az AR. Drone 2.0 eszköz hálózati beállításai

Ezen repülőgép esetében a helyes konfigurációt egyszerűbb volt felállítanom. A Virtual-Reality Peripheral Network egy platform független programcsomag, mely megvalósítja a kapcsolatot a hardveres és szoftveres eszközök között. Az interfészként működő rendszer képes kezelni a ROS által támogatott topic orientált kommunikációt, valamint a OptiTrack szoftverének a Motive rendszernek az adatait. A kapcsolat Internet Protokoll típusú, és Szerver - Kliens alapú hálózatot valósít meg, ahol esetemben a szerver szerepét a Motive tölti be, a kliens pedig a Robot Operating System. Egy adott objektumra vonatkozóan a pozíciós, az orientációs, valamint a gyorsulási adatokat képes közvetíteni. A munkám során a kommunikációt a *vrpn_client_ros* nevű csomaggal valósítottam meg, amely sajnálatos módon a gyorsulásra vonatkozó adatokat nem képes kezelni. Ez nem jelentett nagy problémát, mivel az én kutatásom szempontjából a gyorsulási adatok ismerete nélkül is képes voltam végrehajtani a feladatokat. A csomag segítségével definiált topic egységek a következők:

- **/vrpn_client_node/ardrone/pose**
  A repülőgép pozíció és orientáció adatai
- **/vrpn_client_node/destination/pose**
  Az általam meghatározott cél pozíció adatai

A kommunikációban a már említett PoE funkcióval felszerelt switch is fontos szerepet játszott. A kamerák mellett a Motive és a ROS rendszereket futtató számítógépek is csatlakoztak ehhez az eszközhöz, és a tényleges fizikai kommunikáció ezen keresztül valósult meg.

3.4.2. A Ryze Tello Edu eszköz hálózati beállításai

Ezen repülőgép esetében a tesztrepülések első szakaszában a NatNet elnevezésű programcsomagot használtam. Az OptiTrack cég által létrehozott, és kimondottan erre specifikált modul több lehetőséget biztosít a felhasználójának, mint a korábban említett társa. Szintén Szerver - Kliens architektúrát valósít meg, Internet Protokoll típusú Multicast csomagtovábbítási módszerrel. A ROS rendszerben *natnet_ros* elnevezésű csomagként megtalálható.

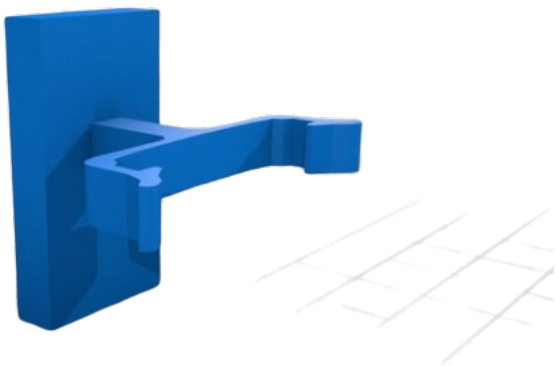
Erre a programcsomagra azért volt szükség, mert a Tello kis méreteiből adódóan a már említett rigid body nevű test, amely a 3.5. ábrán látható, nem fért el a repülőgépen. A megoldást az jelentette, hogy egyetlen markert szereltem a drón tetejére. Sajnálatos módon a VRPN csomag minimum 3 darab markerből álló csoportot képes továbbítani, így a NatNet jelentette az egyetlen lehetőséget. Az egyetlen hátrány, hogy egy marker esetén a Motive nem képes az orientáció adatainak előellítására, de a munkám elvégzése szempontjából ez nem is volt feltétlenül szükséges. A csomag segítségével definiált topic egységek a következők:

- **/mocap/markers/leftovers**
  A repülőgép pozíció adatai

- **/mocap/rigid_bodies/destination/pose**
  Az általam meghatározott cél pozíció adatai

A tesztrepülések alatt többször is előfordult, hogy a NatNet csomag néhány sikeres repülés után, nem volt képes továbbítani a Motive adatait. Ezekre a hibákra nem tudtam logikus magyarázatot találni, így egy új ötletre volt szükség.

A megoldást egy 3D nyomtatással előállított elem jelentette, amelyre a markercsoport testét fel tudtam helyezni. Így a VRPN használatával hibamentesen nyomon követhettem a repülőgép mozgását. Az elem a 3.11. ábrán látható.



3.11. ábra. A végső megoldást jelentő, Tello repülőgépre szerelt, univerzális platform. [36]

3.5. Tesztkörnyezet tervezése

Ebben az alfejezetben az eljárások tesztelésére berendezett környezetet ismertetem. Először a kamerarendszer pontos elrendezését, majd pedig a drónok autonóm repüléséhez készített akadálypályát mutatom be.

A tesztelésre a legalkalmas nyílt beltéri helyiség, az egyetem tornaterme volt. Ez a közel 50 négyzetméteres terem, mely emellett hatalmas belmagassággal is rendelkezett, tökéletes környezetet biztosított a repüléshez.

3.5.1. Kamerák elhelyezkedése

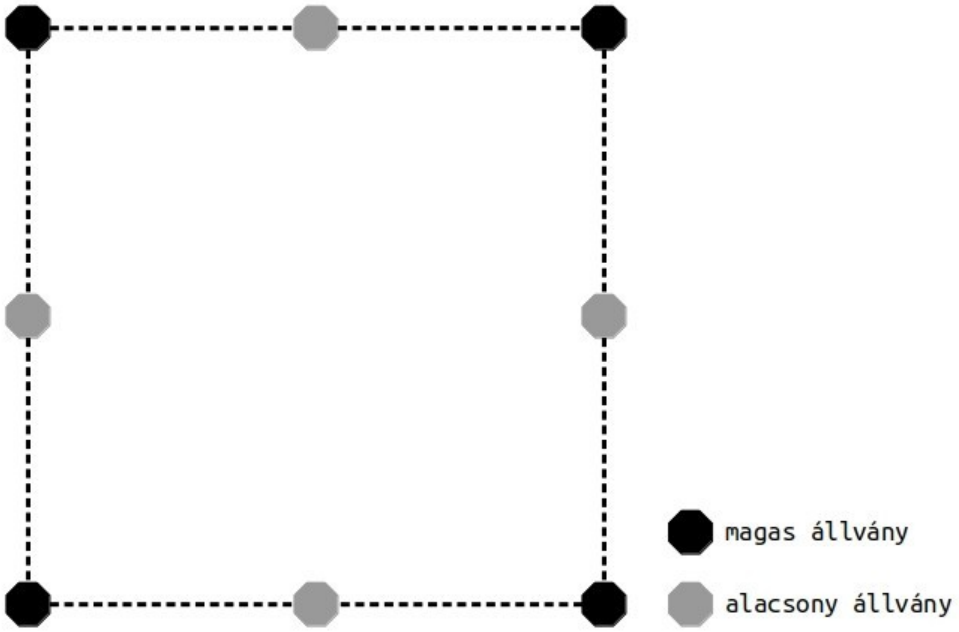
A terem adottságai miatt, a kamerák helyes elrendezésére több lehetőségem is volt. Arra próbáltam törekedni, hogy minél nagyobb teret képesek legyenek megfigyelni, valamint az akadálypáya elemei se takarjanak ki területeket.

A kamerákat kimondottan erre a célra kifejlesztett állványokon helyezkedtek el, melyeketből négy darab akár 6 méteres magasságra is beállítható. Az összesen nyolc kamerából álló rendszer elrendezése akkor volt a legoptimálisabb számomra, ha egy négyzet alapú területet foglal egybe. A négyzet sarkaiban lévő kamerák 4 méteres magasságban helyezkedtek el, és 45 fokos szöget zártak közre a talajjal. Az alacsonyabb állványokra szerelt kamerák egyenként foglaltak helyet a magasak között, és 90 fokos szöget zártak közre a talajjal. Természetesen megpróbáltam a lehető legpontosabban mindegyik kamera képét a négyzet alapú tér középpontjához igazítani. Az állványok egymástól 7 méteres távolságra helyezkedtek el, így mintegy 25 négyzetméteres alapterületű és megközelítőleg 3 méteres magasságú repülési zónát értem el. Az elrendezés alaprajza a 3.12. ábrán látható.

3.5.2. Akadálypálya felépítése

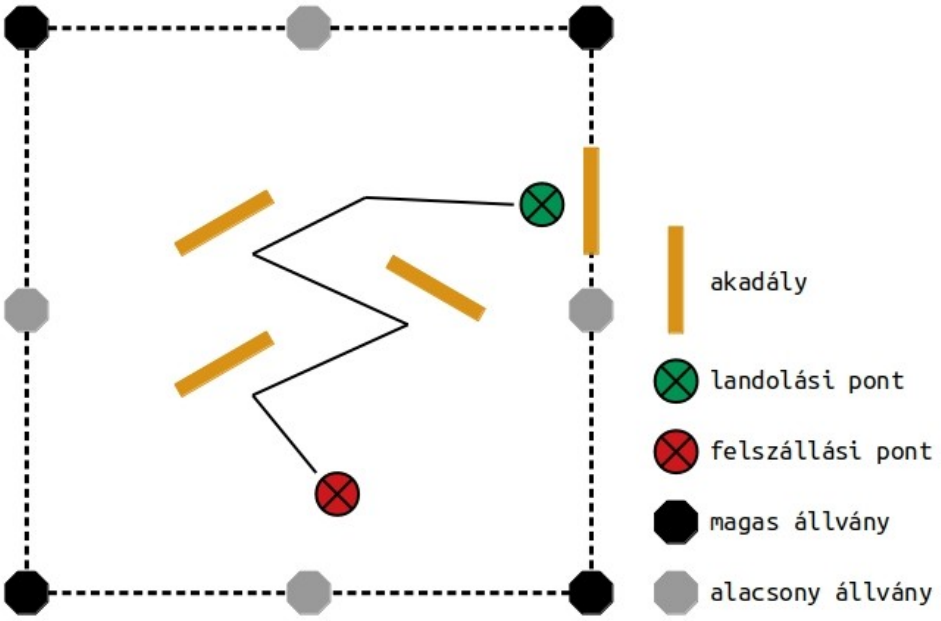
Az akadálypálya megtervezése során arra törekedtem, hogy a két különböző eljárást ugyanazon pályán tudjam tesztelni, így az AR. Drone 2.0 méreteihez kellett igazodnom. Az egyes akadályok között 2-3 méteres távolságot hagytam a repülőgépek biztonságos mozgása érdekében.

Az akadályokat a karon található paravánok képezték. Ezekre szereltem fel az aktuális eljáráshoz szükséges képi adatokat, melyek különböző információkat tartalmaztak a repülő számára az akadály kikerülésére vonatkozóan. Az utolsó képi adatot a tornaterem



3.12. ábra. Az állványok elhelyezkedésének alaprajza.

falán helyeztem el. Erre azért volt szükség, mert a 3 darab paravánból képzett akadály, így is sok esetben kitakarta a drónok tervezett repülési pályáját. Ezt úgy próbáltam ellensúlyozni, hogy a kezdeti 1 méteres repülési magasságot 2 méteresre megnöveltem, ezáltal a magasan elhelyezett kamerák jobban képesek voltak nyomon követni a drónok helyzetét. A kamerák elhelyezkedésének alaprajzát kiegészítve, a 3.13. ábrán szemléltetem az akadálypályát és a tervezett repülési pályájukat is.



3.13. ábra. A tesztrepülés szemléltetésére szolgáló alaprajz.

3.6. Eljárások implementációjának bemutatása

Az eljárások implementálására a Python nyelvet választottam, mely napjainkban egyre divatosabb, valamint robotikai alkalmazások fejlesztésénél is igen elterjedt. Korábban nem foglalkoztam szkript nyelvekkel, így a fejlesztés során törekedtem az egyszerűségre. Hamar megtapasztaltam az ilyen típusú nyelvek nagy előnyét, miszerint a futtatáshoz elegendő egy interpreter használata. A rospy sok hasznos funkcióval van felszerelve, melyeket munkám során is megtapasztaltam. A kutatás során két ROS csomagot hoztam létre, melyek a két eljárás implementációját tartalmazzák.

3.6.1. Szövegfelismerő eljárás implementálásának bemutatása

Program implementálása, valamint integrálása a navigációs rendszerbe, csak részben volt sikeres. Ahogy az eljárás bemutatásánál említettem, a repülőgép a számára információt hordozó szöveges ablakot nem pontosan derékszögben látja. Ez a karakterek értelmezését nagyban megnehezíti, mivel az egyes elemek különböző szögekből egészen más alakúak és gyakran egymásra is hasonlítanak. A probléma lehetséges megoldásai idő hiányában, csak elméleti úton vannak bemutatva, a következő fejezetben.

A program felépítése megegyezik a következő pontban leírt eljárás implementációjával, azzal a különbséggel, hogy esetenként más csomagok kerültek felhasználásra, melyekről szintén tettem említést, korábban az eljárás bemutatása során.

3.6.2. QR-kód felismerő eljárás implementálásának bemutatása

Az implementálás első fázisaként egy launch fájlt hoztam létre. Ez felelős a csomag által használt egyéb más modulok elindításáért, melyek ebben az esetben a VRPN, és ZBAR csomagok, továbbá a repülőgép kezdeti paramétereinek beállítása is itt történik.

A launch fájl működésének tesztelése után kezdődhetett az implementálás folyamata. A fejlesztés során próbáltam olyan változóneveket választani, melyek segíthetik a kód értelmezését. Ez ugyan nem a legszebb megoldás, valamint a PEP8 által előírt szabályok betartása is lehetetlenné vált, mégis segíti a kód megértését.

A programot egy osztályba szervezve hoztam létre, ami a FlyController nevet kapta. Az osztály első soraiban a változók inicializálása mellett, a topic feliratkozásokat és a publikációkat végeztem el. Ezen elemek szintaktikáját egy-egy példán keresztül mutatom be. Ezek a következők:

- *rospy.Subscriber('/tello/camera/image_raw', Image, self.image_callback)*
  - A függvény első bemenete, azaz a *'tello/camera/image_raw'*, az adott topic nevét adja meg, amelyre feliratkozik.
  - A második bemenet a feliratkozás során kapott adat típusát adja meg, mely ebben az esetben az *Image*.
  - A harmadik bemenet egy függvényt hív meg, amely a feliratkozásból származott adatokat kezeli. Ez az úgynevezett *callback* függvény.
- *rospy.Publisher('/tello/cmd_vel', Twist, queue_size=1)*
  - A függvény első bemenete, azaz a *'tello/cmd_vel'*, az adott topic nevét adja meg, ahová a publikáció során adat érkezik.
  - A második bemenet a publikálás során szolgáltatott adat típusát adja meg, mely ebben az esetben a *Twist*.
  - A harmadik bemenet, azaz a *queue_size=1*, a buffer méretét adja meg a publikációkhoz.

A forráskód logikai egységeit a ROS szemantikája alapján callback függvényekbe szerveztem. Ezen függvények kerülnek meghívásra a topic feliratkozások során. Az egységek a következők:

- A *command_ callback* függvény az autonóm vezérlésért felelős egység. Itt történik a különböző QR-kódok által kiváltott parancsok végrehajtása, melyeket a *barcode_ callback* függvény adatai alapján képes megvalóstani.

- Az előbb említett *barcode_ callback* függvény a QR-kódok értelmezéséért felelős, amit a ZBAR csomag segítségével végez. A beérkező adatok String formátumúak, amiket magam rendeltem az egyes QR-kódokhoz a megalkotásuk során.

- Az *image_ callback* egység a drón kamera képét szolgáltatja egy ablak formájában, melyet a openCV nevű könyvtárral oldottam meg. Ebben a függvényben felszereltem a programot egy egyszerű kontrollerrel is, így a repülőgép irányítása a billentyűzet segítségével is lehetséges. A kontroller használati útmutatója a ROS csomagban lévő README fájlban megtalálható, ezenfelül a már említett rospy, konzol üzenetein keresztül is segítséget kap a felhasználó.

- A *tello_ pose_ callback* függvény a Motive által szolgáltatott adatokat dolgozza fel, melyek a repülőgép aktuális helyét adják meg. Ez a NatNet, és a VRPN csomag felhasználásával is elvégezhető. A különbség csak annyi, hogy a VRPN esetében PoseStamped típusú adatok érkeznek, a NatNet esetében pedig MarkerList típusú.

- A *destination_ pose_ callback* függvény felelős az esetleges célpozíció, helyének meghatározásáért. Ezek az adatok az előbb említett módon érkeznek, és mindkét esetben PoseStamped típusúak.

- A *status_ callback* függvény a repülőgép aktuális státuszát biztosítja, mely megadja, hogy a gép éppen landol, tétlen állapotban van, vagy felszállást hajt végre. Más információkat is hordoz ez az adat, de a tesztelés szempontjából ezek nem voltak relevánsak.

- További függvények, a repülőgép irányításra vonatkozó konkrét utasításokat végzik, melyeket a *command_ callback* egység az autonóm repülést, *image_ callback* egység pedig a kontrollert működtetését valósítja meg.

3.6.3. Verziókövetés

A munkám során a git verziókövető rendszert használtam. Ennek segítségével nyomon követhető a forráskódok egyes verziói, valamint a további fejlesztések is könnyedén megvalósíthatóak a branch funkció igénybevételével. A projekt forráskódjai, valamint ezen dokumentum a https://github.com/mihalymokos/sak GitHub oldalon érhető el.

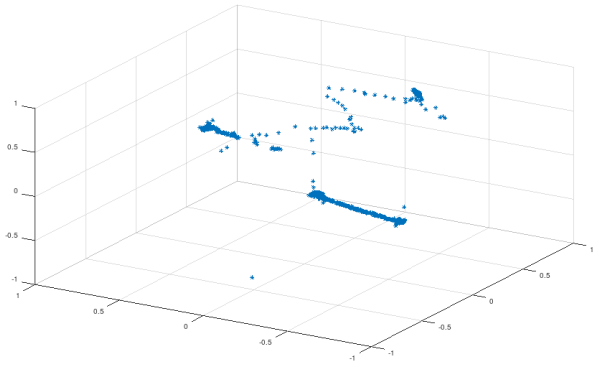
4. fejezet

Eredmények

A kutatásom során próbáltam arra törekedni, hogy átfogó ismereteket szerezzek minden olyan technológiáról és eszközről, melyeket munkámhoz részben, vagy egészben használtam. Ebben a fejezetben szeretném összefoglalni a munkám során szerzett tapasztalataimat és eredményeimet, valamint néhány továbbfejlesztési lehetőségek is bemutatok.

4.1. A tesztrepülések eredményei

A tesztrepülések a előző fejezetben leírtak alapján történtek. A drón sikeresen hajtotta végre az autonóm átrepülést az általam épített akadálypályán. A repülést az OptiTrack rendszerrel követtem nyomon. A Motive szoftverből kiinportáltam a repülés adatait, melyet a GNU Octave nevű program segítségével ábrázoltam. Az akadályok miatt sokszor a repülőgép takarásban volt, ezáltal a drónra szerelt markercsoport nem volt látható a kamerák számára. A érzékelt pontok háromdimeziós térben való ábrázolása az 4.1 ábrán látható.



4.1. ábra. A Motive szoftver által rögzített repülési adatok ábrázolása, GNU Octave program segítségével a háromdimenziós térben.

A tesztrepülések videofelvételei, a hozzájuk tartozó forráskódok, valamint ezen dokumentum a https://dev.itk.ppke.hu/uavVisionLab/tello_qr oldalon érhetők el.

4.2. A kutatás során szerezett tapasztalatok

Úgy érzem, hogy az általam választott feladat már a kutatás kezdetétől fogva méltó kihívások elé állított, melyek megoldása olykor igen nehézen ment, mégis kijelenthetem, hogy élvezettel végeztem a munkámat.

A munkám első fázisa a szakirodalom tanulmányozása és a hasonló alkotások megismerése volt. Úgy gondolom, hogy kellő ismeretet szereztem az alkalmazni kívánt technológiák területén, továbbá a munkám során felhasznált eszközökkel és azok kezelésével kapcsolatban is. Fontos része volt ez a projektemnek, mivel itt kaptam meg azokat az alapismereteket és ötleteket, melyekből felépíthettem a saját kutatásom.

A tervezés és fejlesztés szakaszában sajnos sok, előre nem várt problémával találkoztam. Gondok akadtak a különböző rendszerek kompatibilitásából adódóan, továbbá az egyes ROS csomagok forráskódjai is hibásak voltak. Ebből kifolyólag számos olyan probléma megoldására volt szükség, melyekkel a korábbiakban még nem találkoztam, így ezek megoldása során sok új tapasztalatot szereztem. A helyes hálózati beállítások megalkotásában is új ismeretekre tettem szert, amik a jövőbeli kutatásaimhoz nagy segítséget nyújthatnak. A harverközeli problémák kezelésében is sok új tapasztalatot szereztem, melyek számomra eddig lehetetlen küldetéseknek bizonyultak.

A tesztelés fázisában érzékeltem, hogy mennyire nagy szerepe van a fizikának a gyakorlati életben. A korábban végzett szimulációs tesztekkel szemben, amik erősen determinisztikusak voltak, itt a valós világ tényezőivel kellett számolnom, melyek egy igen más szemléletmódot követelnek meg. Az első tesztrepülések esetén nem figyeltem eléggé a repülőgépek paramétereire vonatkozó kritikus pontokra, valamint a repülőgép és az akadályok egymásra mért erőhatásaira. Ezeket a drónok lassabb mozgásával és időnként akár több másodperces egy helyben való várakozásával oldottam meg. Egy elegánsabb lehetőség lett volna, ha drón további szenzorait használom ezen jelenségek ellensúlyozására, de a projektem célja nem a sebesség volt, így idő hiányában a kutatás relevánsabb tényezőire koncentráltam.

A munkám dokumentációjának elkészítése közben szerzett tapasztalataimat is fontos kiemelnem. Korábban nagy félelemmel gondoltam arra, hogy a jövőben egy ilyen méretű alkotást kell készítenem. Ezen kutatás dokumentálása után kevésbé riadok vissza egy hasonló volumenű feladattól. A szövegalkotásra vonatkozó új ismeretek ugyan nem kimondottan ebben a projektben nyilvánulnak meg, az esetleges jövőbeli alkotásaimra vonatkozóan nagy szerepet játszanak.

Általánosságban azt mondhatom, hogy rengeteg tapasztalatot és új ismeretet szereztem a kutatásom során, melyek segítik a jövőbeli pályafutásomat.

4.3. A kutatás kritikai elemzése

A kutatás során több olyan tévedést is elkövettem, melyek rontották a kutatás minőségét. Ezen hibák közül az első és legfontosabb az időbeosztás. A munkám kezdeti fázisában túlzottan sok időt töltöttem a szövegfelismerő eljárással való autonóm vezérlés megalkotásával. Ezek után nem maradt elég idő egy másik, egyszerűbb módszer implementálásra, vagy a fennálló probléma megoldására.

További időbeosztásból adódó hibák találhatóak, a QR-kód felismerő eljárás implementálásában, ahol a tervezettel ellentétben az OptiTrack kamerarendszert, nem hatékonyan használom. Egy további kritikus pont, a vészhelyzet esetén történő leállás. Erre akkor kerülhet sor ha repülőgép az autonóm repülés során valamilyen okból kifolyólag elhagyja a kamerarendszer által megfigyelt területet. Az ilyen esetek elkerülésére szolgál az egyszerű kontroller, amellyel bármikor lehetőség van landolásra utasítani a gépet. A esetleg további balesetek elkerülésére két konzol parancs is a rendelkezésre áll a manuális leállításra, mely a következő:

```
$ rostopic pub /tello/land std_msgs/Empty "{}"
$ rostopic pub /tello/emergency std_msgs/Empty "{}"
```

Az első parancs landolásra utasítja a drónt, továbbá letiltja a futtatott programból kapható összes műveletről. A második sorban látható parancs is hasonlóan működik, azzal a különbséggel, hogy ez a drón összes propellerét leállítja, így az nem tehet kárt a környezetében. Ez persze azzal a mindenki által jól ismert jelenséggel jár, hogy a repülő a gravitáció hatására a földre zuhan.

Egy fontos tervezési hiba, melyre nem találtam egy jó megoldást sem, miszerint a ROS csomagokban használt másik modulok fejlesztés alatt állnak, így idővel kompatibilitási problémák merülhetnek fel. Ezen probléma elkerülésére, a használt modulok forráskodjainak átültetésével lett volna lehetőség. Sajnos ebben az esetben is felléphetett volna inkompatibilitás, csak itt a ROS disztribúciója és az én csomagok között.

4.4. Továbbfejlesztési lehetőségek

A munkám során több olyan ötletem is támadt, melyek jó alapot jelenthetnek egy későbbi kutatáshoz számomra, vagy akár más hasonló témával foglalkozó hallgató számára. Ebben az alfejezetben ezeket szeretném röviden bemutatni.

4.4.1. Szövegfelismerő eljárás helyreállítása

Az első és legfontosabb fejlesztési lehetőség az általam készített szövegfelismerő eljárás helyreállítása. Egy jó alternatíva erre az algoritmus tanítóhalmazának bővítése. A felismerendő karakterekhez több példát szükséges megadni, melyek más-más szögből látják az adott elemet, így sokkal biztosabb találati eredmény érhető el.

4.4.2. Összetett akadálypálya építése

A második ötletem egy összetett akardálypálya építése és a repülő ezen való átrepülése. Ebben az esetben több kamera alapú információ kinyerésére alkalmas eljárással lenne felszerelve a repülő, melyek mindegyikére szüksége van az akadályok elkerüléséhez. Ez egy bonyolultabb és több munkát igénylő feladat, de jó alapja lehet egy komolyabb kutatásnak.

4.4.3. Swarming

A swarming szó sokak számára ismeretlen lehet. Magyarra fordítása rajzást jelent, ami hagyományos értelemben a méhcsaládok szaporodásának folyamata. A méhcsaládokat szuperorganizmusoknak tekintik a tudósok, ami rövid azt jelenti, hogy igen magas fokú munkamegosztásban élő egyedekből álló szervezet.

A swarming jelentése a robotikában és a pilóta nélküli repülőgépek esetében vett jelentése, hogy több robot vagy repülőgép egy közös cél érdekében egyszerre hajtanak végre feladatokat, melyekhez összedolgoznak. Ezek a feladatok egészen különbözőek lehetnek, és optimális esetben az eszközök az általuk legjobban teljesíthető feladatokat hajtják végre. A technológia széleskörben alkalmazott a hadászatban, valamint léteznek már olyan vízi drónflották, melyek közös együttműködéssel gondoskodnak a fürdőzők biztonságáról a szabadstrandok vizeiben.

A második teszt, azaz a QR-kód detekció során használt Ryze Tello Edu repülőgép kimondottan jó választás lehet egy ilyen kutatáshoz. A kis méret itt nagy előnyt jelenthet, mivel így a drónok nem zavarják egymást a feladatok végrehajtásában és a manőverezésben.

5. fejezet

Összefoglalás

A kutatásom bemutatása után nem maradt más, minthogy összegezzem a dolgozatban leírakat. Ebben a fejezetben rövid leírást adok az eddig olvasottakról.

A munkám első lépéseként a pilóta nélküli repülőgépekkel és a beltéri navigációs rendszerekkel ismerkedtem meg. Az általános tudnivalók után az egyetem UAV laboratóriumában megtalálható drónok paramétereit térképeztem fel. Ezt követően a szakirodalmat, és a téma hasonló alkotásait tanulmányoztam. Több olyan projekttel is találkoztam, melyek némi tapasztalatot és jó kiindulási alapot jelentettek a saját kutatásom megtervezéséhez.

A következő lépés az általam használni kívánt eszközök beható tanulmányozása volt. Itt teljes képet kaptam a kiválasztott drónok paramétereiről, továbbá vezérlésüket is elsajátítottam. Megismerkedtem az OptiTrack kamerarendszer elemeivel és a hozzájuk tartozó Motive nevű szoftverrel. Megtanultam a Robot Operating System használatát, és a keretrendszer moduljainak kezelését. A ROS és a Motive kommunikációjának felállításához további eszközök alkalmazásával ismerkedtem meg.

Az fejlesztés során egy sikeresen működő ROS csomagot hoztam létre, mely megvalósít egy akadálypályán való autonóm átrepülést pusztán képi adatok alapján. Az csomag kapcsolatot létesít az OptiTrack kamerarendszerrel, így az onnan származó adatok apján is lehetőség nyílik a repülőgép vezérlésére. A csomagot felszereltem egy egyszerű kontrollerrel a drón irányítására vonatkozóan. Számos tesztet végeztem, melyeken keresztül megtapasztaltam a rendszerem korlátait és kritikus pontjait.

Felhasznált irodalom

[1] *Gesture Controls | BMW Genius How-To*. elérhető: https://www.youtube.com/watch?v=wqvAPskg_k0 (elérés dátuma 2019. 11. 28.).

[2] *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*. English. S.l.: IEEE., 2018, OCLC: 1096822096, ISBN: 978-1-5386-5241-1.

[3] *How drones are changing the rice industry*, en. elérhető: https://japantoday.com/category/tech/how-drones-are-changing-the-rice-industry (elérés dátuma 2019. 12. 14.).

[4] *Amazon.com: Prime Air*. elérhető: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011 (elérés dátuma 2019. 12. 14.).

[5] J. Bown, „The snow patrol drones saving skiers from an icy death“, en-GB, *BBC News*, 2019. febr. elérhető: https://www.bbc.com/news/business-47309085 (elérés dátuma 2019. 12. 14.).

[6] Y. Tiumentsev, *Neural network modeling and identification of dynamical systems*, 1st edition. Cambridge, MA: Elsevier, 2019, ISBN: 978-0-12-815254-6.

[7] I. Jelic és D. Vrkic, „QR codes in library - Does anyone use them?“, 2013. jan., 695–699. old., ISBN: 978-953-233-076-2.

[8] *QR-kód*, hu, Page Version ID: 21699400, 2019. szept. elérhető: https://hu.wikipedia.org/w/index.php?title=QR-k%C3%B3d&oldid=21699400 (elérés dátuma 2019. 12. 1.).

[9] *OptiTrack - Motion Capture Systems*. elérhető: https://optitrack.com/ (elérés dátuma 2019. 11. 24.).

[10] *ROS.org | powering the world's robots*. elérhető: https://www.ros.org/ (elérés dátuma 2019. 11. 24.).

[11] G. Nagy, T. Nartker, és S. Rice, „Optical character recognition: an illustrated guide to the frontier“, *Proceedings of SPIE - The International Society for Optical Engineering*, 3967. vol. évf., 58–69. old., 1999. dec. DOI: 10.1117/12.373511.

[12] J. Lee, J. Wang, D. Crandall, S. Sabanovic, és G. Fox, „Real-Time, Cloud-Based Object Detection for Unmanned Aerial Vehicles“, en, *2017 First IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan: IEEE, 2017. ápr., 36–43. old., ISBN: 978-1-5090-6724-4. DOI: 10.1109/IRC.2017.77. elérhető: http://ieeexplore.ieee.org/document/7926512/ (elérés dátuma 2019. 11. 17.).

[13] P. Benedek, „Kamera alapú beltéri navigáció az AR drone eszközön“, hu, 40. old.,

[14] H. R. Oosterhuis és A. Visser, „Evaluation of tracking algorithms to autonomously following dynamic targets at low altitude“, en, 10. old.,

[15] Z. Kalal, K. Mikolajczyk, és J. Matas, „Tracking-Learning-Detection“, en, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34. vol. évf. 7. sz. sz., 1409–1422. old., 2012. júl., ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2011.239. elérhető: http://ieeexplore.ieee.org/document/6104061/ (elérés dátuma 2019. 11. 17.).

[16] *Parrot AR.DRONE 2.0 Power Edition*, en-us, 2017. nov. elérhető: https://www.parrot.com/us/drones/parrot-ardrone-20-power-edition (elérés dátuma 2019. 11. 20.).

[17] *Tello*. elérhető: https://www.ryzerobotics.com/tello (elérés dátuma 2019. 11. 20.).

[18] *Drone Tello EDU DJI - Educational Drone to be programmed*. elérhető: https://www.robot-advance.com/EN/art-drone-dji-tello-edu-2610.htm (elérés dátuma 2019. 11. 28.).

[19] *Documentation - ROS Wiki*. elérhető: http://wiki.ros.org/ (elérés dátuma 2019. 11. 25.).

[20] T. Tamás, „Robot Operating System - ROS segédlet“, 25. old.,

[21] *Vicon | Award Winning Motion Capture Systems*. elérhető: https://www.vicon.com/ (elérés dátuma 2019. 11. 24.).

[22] *Power over Ethernet (POE) Explained - Understanding and using POE*. elérhető: https://www.veracityglobal.com/resources/articles-and-white-papers/poe-explained-part-1.aspx (elérés dátuma 2019. 11. 24.).

[23] *Prime 13 - technical specifications*, OptiTrack. elérhető: http://optitrack.com/products/prime-13/specs.html (elérés dátuma 2019. 11. 24.).

[24] *Accessories*, OptiTrack. elérhető: http://optitrack.com/accessories/index.html (elérés dátuma 2019. 11. 24.).

[25] *Motion Capture Markers*, en. elérhető: http://optitrack.com/products/motion-capture-markers/index.html (elérés dátuma 2019. 12. 7.).

[26] *Calibration tools*, OptiTrack. elérhető: http://optitrack.com/products/tools/index.html (elérés dátuma 2019. 11. 24.).

[27] *Linksys WRT1900AC AC1900 Dual-Band Wi-Fi Router*, en_EU. elérhető: http://www.linksys.com/pl/p/P-WRT1900AC/ (elérés dátuma 2019. 12. 1.).

[28] G. Papp-Szentannai és G. Kovács, *DroneProject/DroneMainRepo*, original-date: 2015-07-29T13:42:01Z, 2019. máj. elérhető: https://github.com/DroneProject/DroneMainRepo (elérés dátuma 2019. 11. 18.).

[29] *Decoding CAPTCHA's | Ben E. C. Boyter*. elérhető: https://boyter.org/decoding-captchas/#recognition (elérés dátuma 2019. 11. 18.).

[30] W. K. Pratt, *Digital image processing: PIKS inside*, 3rd ed. New York: Wiley, 2001, ISBN: 978-0-471-37407-7.

[31] *Up and flying with the AR.Drone and ROS: Getting started | Robohub*. elérhető: https://robohub.org/up-and-flying-with-the-ar-drone-and-ros-getting-started/ (elérés dátuma 2019. 12. 3.).

[32] B. Eck, *aeriu*, en. elérhető: https://aeriu.co/ (elérés dátuma 2019. 11. 28.).

[33] Z. András, *Drónokkal teszi pofonegyszerűvé a leltározást egy magyar startup*, hu, 2018. jan. elérhető: https://kosarertek.hu/technologia/dronokkal-teszi-pofonegyszeruve-a-leltarozast-egy-magyar-startup/ (elérés dátuma 2019. 11. 28.).

[34] *Augmented Reality / 3D Tracking | Welcome*. elérhető: http://virtual.vtt.fi/virtual/proj2/multimedia/index.html (elérés dátuma 2019. 12. 1.).

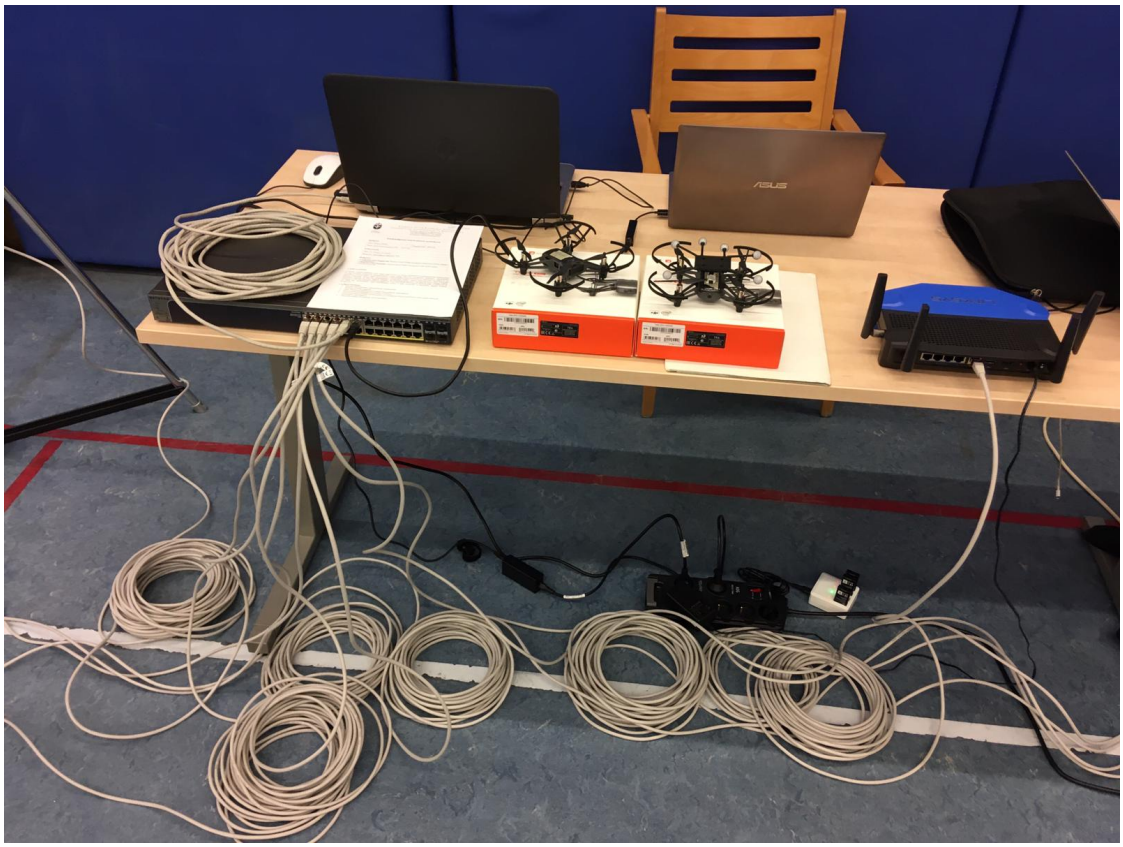
[35] *ViSP*, en-US. elérhető: https://visp.inria.fr/ (elérés dátuma 2019. 12. 1.).

[36] *DJI Ryze Tello universal platform by Yugen42 - Thingiverse*. elérhető: https://www.thingiverse.com/thing:3023822 (elérés dátuma 2019. 12. 7.).

A. függelék

Melléklet

Ebben a kiegészítő fejezetben gyűjtöttem össze mindazt, ami nem szükséges a dolgozat megértéséhez, de segíti azt.



A.1. ábra. Az úgynevezett vezérlő állomás.

A.2. ábra. Az első három akadály, a QR-kódokkal felszerelve.

A.3. ábra. A falon látható, utolsó akadály.

A.4. ábra. A tesztelések során használt magas állvány.

A.5. ábra. A tesztelések során használt magas állvány.