Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Szakdolgozat

Kamera alapú beltéri navigáció az AR drone eszközön

Petrovicz Benedek
Mérnök informatikus BSc
2017

Témavezető:
dr. Zsedrovits Tamás

Alulírott Petrovicz Benedek a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának hallgatója kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem és a szakdolgozatban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem. Ezt a Szakdolgozatot más szakon még nem nyújtottam be.

_____

Petrovicz Benedek

Tartalomjegyzék

Kivonat

A dolgozat arra törekszik, hogy az automatizált drónvezérlés témában érdekelt személyeknek, eligazítást és kiindulási alapot adjon. Amellett, hogy elméleti síkon, általános áttekintést ad a témáról, mellékletként tartalmazza ezen gondolatok tényleges megvalósítását is.

A kidolgozás kifejezetten olyan drón kutatókat céloz, akik pontos méréseket szeretnének végezni és ezáltal hatékony algoritmusokat fejleszteni. A kutatás során felhasznált összes eszköz, fontos részét képezi a rendszernek, mivel bármelyik elhagyása esetén használhatatlanná válik a kidolgozott kód. Bár az eszközök szimulálására van lehetőség, de az nagy mértékben ronthatja a mérések eredményét.

A kutatás során felhasznált drón a Parrot cég AR.Drone 2.0 nevű terméke. Viszonylag kis térben is biztonságos körülmények között használható. A biztonságot maga a drónra szerelt kemény szivacs alapú keret adja, mely amellett, hogy védi a drón propellereit, azt is megakadályozza, hogy a környezetében nagyobb kárt tudjon tenni. További biztonságot nyújt a drón fedélzeti számítógépén futtatott szkript, ami azonnal leállítja a drón rotorjait, amint azt érzékeli, hogy az eszköz elérte a 90 fokos dőlésszöget, ami által könnyedén megállítható az akkumulátorhoz való hozzáférés nélkül is.

A felhasznált eszközök közül a második legfontosabb, az OptiTrack cég mozgáskövetésre alkalmas kamerarendszere. Ez képes akár centiméteres pontossággal lekövetni és háromdimenziós térben ábrázolni lényegében bármit, ami fel van szerelve a kamerákhoz kidolgozott fényvisszaverő markerekkel.

Az eddig felsorolt eszközök kommunikációját pedig, egy Linux rendszerrel ellátott gépen futó Robot Operating System (ROS) teszi lehetővé. Ezt a nyílt forráskódú rendszert széles körben alkalmazzák a robotikával foglalkozó cégek és kutatók, így rengeteg eszközhöz érhetők el különböző felhasználásra szánt implementációk.

A repülés közbeni utasítás közvetítést, QR kódok és a drón 2 kamerája segíti elő. A jelek felismerését a kamera képéből az ar_track_alvar nevű ROS csomag valósítja meg. Ez a csomag kifejezetten jól működik gyenge felbontású kamera remegő képével is, amiből nincs hiány egy ilyen projekt során. Az egyetlen hátrány abból adódik, hogy a drón egyszerre csak az egyik kamerája képét tudja közvetíteni a rendszernek.

A megvalósított forráskód egy ROS csomag formájában használható fel. Fel van készítve a rendszer minden összetevőjének vezérlésére és az adataik feldolgozására. A felhasználónak csupán az eszközök IP címeit kell pontosan megadnia a csomag számára, illetve, hogy drón melyik kameráját szeretné használni a QR kódok felismerésére.

A QR kódon szereplő adat alapján több különböző előre megírt algoritmust is képes végrehajtani. A projektben csak előre kiszámolt animációkat kapott meg parancsként a drón, amelyek segítségével képes átrepülni vagy egy irányban kikerülni az előtte lévő akadályt.

További fejlesztések során már csak ezzel a részével kell foglalkoznia az adott kutatónak. További szenzorok adatainak a felhasználása is egyszerűen megoldható. Mivel a kód python nyelven készült, gyorsan és problémamentesen lehet tovább alakítani.

Abstract

The thesis pursues at giving a guidance and a starting-point to people who are intrested in automated drone controlling. Besides that it gives a general theoretical overview of the topic, it contains the implementation of these thoughts as an attachment.

This work mostly aims at those drone researchers, who want to take accurate measurements and to develop effective algorithms with them. All of the items used in this research are important parts of the system, because leaving out any of them makes the code unusable. Although simulating them is another possibility, but it spoils the measurement results greatly.

The drone used in this research is the product of the Parrot company called AR.Drone 2.0. It can be used in relatively small places, under safe conditions also. The main reason of this safety is the hull of the drone made out of some hard foam. Besides that it protects the propellers of the drone, ensures that the drone is not causing any great damage in the environment around it. A script running on the on-board computer of the drone provides additional security, because it blocks the rotors of the drone as soon as it detects that the drone is tilted by 90 degrees so it can be stopped easily without reaching its battery.

The second most important item used in the research is the camera system of OptiTrack that is capable of motion capture. It is able to track with centimeter accuracy and represent in a three dimensional space essentially anything that is equipped with the reflective markers designed for these cameras.

These two items are communicating through Robot Operating System (ROS) on a computer with Linux. This open-source framework is used widely by companies and researchers intrested in robotics, so there is a wide variety of implementations for a lot of tools and devices.

Giving instructions mid-air, is solved by detecting and decoding QR codes from the video feed of the two camera on the drone. The detection from the camera feed is done by the ar_track_alvar ROS package. This package works really good with shaky image of low resolution cameras, that is frequent in projects like this. The only drawback is, that the drone can broadcast the video feed of only one camera at once.

The implemented source code can be used as a ROS package. It is suitable for controlling all of the components in the system and processing their data. The user is only needed to give the correct IP address of the items, and to choose which camera is used for detecting QR codes.

It is possible to call multiple different algorithms according to the data in the QR code. In the project there were only pre defined animations for bypassing over or next to an obstacle that is before the drone.

In further development processes this part is only intended to be worked on by the researcher. Data of additional sensors can be used easily. As the code has been written in python it can be formed rapidly fast and without any problem.

1. Bevezetés

A kutatás fő célja a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karán elindított UAV laboratórium alapjainak kidolgozása és lefektetése. Rengeteg lehetőség rejlik ebben a gyerekcipőben járó technológiában, így fontosnak tartom, hogy minél előbb megismerjék a hallgatók és részesei lehessenek a fejlesztésének.

A tervezés fő szempontja egy olyan rendszer megvalósítása, ahol az AR drón az általunk írt szkriptek segítségével, automatizált módon képes bizonyos feladatokat végrehajtani. Az OptiTrack kamerarendszer közreműködésével, egy olyan teszt környezetet hozhatunk létre, ami amellett, hogy megbízható, rengeteg plusz információval szolgál a hatékonyság növelésének érdekében. Könnyedén szimulálhatunk vele különböző háromdimenziós terekben való mozgást.

Fontos lépés lehet ez az egyetem életében, mivel a jó alapok megteremtése ebben a témában, nagyon jó lehetőségeket nyithat a jövő hallgatói felé. Aki nem zárkózik el teljes mértékben a közösségi médiáktól, az nagy eséllyel hallott vagy olvasott már drónokról és azok alkalmazásáról szóló cikket. Úgy gondolom ma pontosan abban a világban élünk, ahol ez az egyik leginkább felkapott téma fiatalok és idősebbek között is, mivel olyan piaci területeket nyitott meg a technológia, ami eddig még ismeretlen volt sokak számára.

1.1. Drónok

Sokan nincsenek tisztában a drón mint fogalom pontos jelentésével. A legtöbben a katonai alkalmazásra gondolnak, ahol csapásmérő vagy kémkedő eszközként hasznosítják. Ma már talán többen is vannak, akiknek a szó hallatán a játékra vagy videófelvételre alkalmas 4 rotoros, viszonylag kis méretű eszközök jutnak az eszébe.

A drón eredeti neve az angolban UAV-ként megismert Unmanned Aerial Vehicle vagy magyarul pilóta nélküli légi jármű. Ez kevésbé hangzatos ám annál beszédesebb név. Bármilyen eszközt nevezhetünk drónnak ami alkalmas repülésre úgy, hogy nincs pilóta a fedélzetén. Ebből következik, hogy egy drón bármilyen kicsi vagy nagy lehet, a határokat csak a fizika törvényei szabják meg. A parancsokat általában távvezérléssel kapja valamilyen rádió frekvencián vagy a fedélzetén található szenzorok adatai alapján generálja.

1.2. Alkalmazási területek

Jogos kérdés lehet, a drónok fontossága és gyakorlati haszna vagy egyáltalán ezen projekt értelme. Manapság ezekből egyre többet látni és hallani mind a nagy cégek alkalmazásában mind pedig a halandó kisembereknél is.

Természetesen a legkorábbi fejlesztések olyan problémákat igyekeztek megoldani, mint például a városon belüli csomagszállítás (lásd Amazon), vagy az emberéletek megmentése közben fellépő akadályok legyőzése (lásd hegyimentés). Azt hiszem kijelenthetjük, hogy a fő alkalmazási módokat a nagy multinacionális cégek, a hadsereg, illetve az egészségügy határozza meg, mint ahogy az sok más eszköz esetében is előfordul. Sajnos az is elmondható, főleg a nagy cégek esetében, hogy nem fordítanak elég figyelmet az alkalmazásuk közben fellépő veszélyforrásokra. Sajnos a drónok a legtöbb esetben nem rendelkeznek a megfelelő biztonsági rendszerrel.

Ez felveti olyan algoritmusok és eszközök kidolgozását, amik minél egyszerűbben alkalmazhatók és megelőzik ezeket. Itt jönnek szóba az olyan kutató csoportok, aminek mi is szándékozunk az alapjait lefektetni. Rengeteg alap algoritmus fejlesztésére szükség van, olyanokra, mint például, a virtuális háló, ami nem engedi, hogy a drón elhagyjon egy megadott területet. Természetesen ezekre vannak már megoldások, viszont ezek sok esetben nem felelnek meg azoknak az elvárásoknak, amik szerint nem okozhatnak kárt a környezetükben.

Vannak persze olyan alkalmazási területei is a drónoknak, ahol csekély a környezetre mért károk esélye és mégis hatalmas hasznot tud hozni. Ilyen például a drónok mezőgazdaságban való felhasználása. (cikk) Képes a gazda által birtokolt terület megfigyelésére és különböző adatok elemzésére. Alkalmas például felmérni, hogy melyik területek azok, amik több gondozást igényelnek. Ezen kívül alkalmazható még veteményezésre is, ami igen hasznos lehet nagy területek esetén. Nagy előnye itt a drónoknak, hogy sokkal gyorsabban képesek elvégezni a feladatukat az eddig használt megoldásokhoz képest.

Hasznosítják ezeken kívül még bányáknál a kibányászott nyersanyag mértékének vagy minőségének megállapítására. (cikk) Továbbá nagy ipartelepek esetén biztonsági megfigyeléshez is felhasználhatók. (cikk)

1.3. Robot Operating System

A Robot Operating System vagy rövidebben csak ROS egy nyílt forráskódú, általánosított „operációs rendszer” robotokhoz. Megvalósítja a hardver szintű absztrakciót, képes kezelni a hardverek vezérlését és az egyes folyamatok közti kommunikációt.

A ROS rengeteg olyan eszközt és könyvtárat biztosít, ami lehetővé teszi a különböző platformokon való fejlesztést. Támogatja a kód újra felhasználást, ami nagymértékben megkönnyíti a fejlesztők munkáját. Rendelkezik beépített vizuális megjelenítővel is, ami kiváló segítséget nyújt a teszt környezetünk megjelenítéséhez a kódot futtató gépen. A kutatás során készített forráskód is egy ROS csomagként érhető el.

1.4. Motion Capture

A Motion Capture vagy magyarul mozgásrögzítés, lényegében bármilyen objektum mozgásának a felvételét takarja, ami háromdimenziós térben egy számítógép és sok szenzor segítségével történik. Esetünkben az objektumot fényvisszaverő markerekkel szereljük fel, amiknek a térbeli adatait az OptiTrack kamerarendszer állapítja meg, majd integrálja ezeket az adatokat a saját rendszerébe, amiket utána képes közvetíteni a ROS számára is. A technológia segítségével pontos méréseket és szimulációkat végezhetünk, amik a valós térből származó adatokkal dolgoznak.

2. Feladatkiírás

A szakdolgozathoz a következő feladatok és elvárások lettek előzetesen meghatározva. Ezek alapján követem végig a munka menetét és a teljesítettségi rátáját. Néhány feladatcím általánosabb megfogalmazást kapott, így is biztosítva számomra az alkotói szabadságot. Igyekeztem ezekből is a legtöbbet kihozni.

2.1. Szakirodalom tanulmányozása

A feladatok közé tartozik a drón dokumentációjának (link) és a kapcsolódó szakirodalom tanulmányozása és minél részletesebb megismerése. Ezek elengedhetetlen részei a drónnal végzett hatékony munkának. Ezáltal teljes képet kaphatunk azokról az eszközökről, amiket később hasznosíthatunk a minél gördülékenyebb munkafolyamat eléréséhez.

2.2. Repülő megismerése, irányításának megtanulása

Ez magában foglalja a drón specifikációjának és a kapcsolódó eszközök pontos ismeretét. Ilyen eszköz például az ardrone_autonomy (link) nevű ROS csomag, ami lehetővé teszi a drón irányítását egy számítógépen keresztül. A csomag egy magas szintű nyelvezetet biztosít a kezelője számára. További segítséget nyújthat az ardrone_tutorials (link) csomag ismerete, ami egy további absztrakciós szintet biztosít a drón minél egyszerűbb irányításához.

2.3. Repülési tesztek végzése

Érdemes a munka megkezdése előtt néhány repülési tesztet végezni, mivel ez a kutatás szempontjából legalább annyira fontos, mint az előző pontban ismertetett tudás elsajátítása. Fontos, hogy ezeket a teszteket főleg azon a helyen végezzük, ahol a kutatás későbbi stádiumai is zajlani fognak, mivel a repülőgép karakterisztikájából adódóan más és más nehézségek jelentkezhetnek különböző környezetek esetén.

2.4. QR kódok felismerése a repülő kamerájának segítségével

Mivel a drón, repülés közben QR kódokról kapja az utasításokat, így fontos volt, hogy képes legyen a kamerája képével felismerni, majd dekódolni azokat. Ehhez természetesen el kellett dönteni, hogy saját megoldást készítek egy jól működő algoritmus alapján, vagy keresek egy alkalmas ROS programot, ami fel van készítve a feladat adta nehézségekre. Lévén, hogy a drón legnagyobb felbontású kamerája is csak a piacon elterjedt 720p minőséget hozza a csekély 30 képkocka per másodperc mellett, feladta a leckét, hogy a program képes legyen megbízhatóan felismerni a QR kódokat.

2.5. Mesterséges akadálypálya tervezése és építése

Az akadálypálya tervezésre és építésre azért volt szükség, hogy a kutatás során egy jól definiált környezetben, egyértelmű feladatokat tudjak definiálni, amivel jól mérhető a haladásom mértéke. Így egy kézzelfogható cél lebegett előttem, amit igyekeztem minél precízebben megvalósítani. Ebből adódóan egy másik cél esetén természetesen előfordulhatnak hiányosságok, ezért igyekeztem olyan esetet választani, ami minél általánosabb feladat megoldását hivatott ellátni. Így egyszerűbben átültethető más kutatási projektek esetén az adott kutatók saját környezetébe.

2.6. AR Drone irányítása a repülő szenzoraiból, valamint az Optitrack rendszerből származó információ és ROS segítségével

A sikeres tesztek és tanulmányozás után, a konkrét feladattal a szemem előtt, fontos volt, hogy egy tényleges megoldást is adjak a projektben felmerülő problémákra. Ez lényegében egy fejlesztői folyamat, rengeteg teszteléssel ötvözve. Magában foglalja egy ROS csomag létrehozását és kitöltését a megfelelő kódokkal, amik egy helyen képesek lekezelni a drón szenzoraiból és az OptiTrack rendszerből származó adatok begyűjtését és feldolgozását. Természetesen ennek a feladatnak része volt a kitalált probléma megoldása és a kapcsolódó mérések elvégzése is.

2.7. Elkészült munka és mérések dokumentálása

Fontos, hogy a munka megfelelően legyen dokumentálva, mivel ez az egyetemen egy teljesen új irányvonal az eddigi kutatások között, így bármilyen információ fontos szerepet játszhat a később bekapcsolódó hallgatók számára. Ehhez én összesen négy platformot terveztem igénybe venni, hogy a munka minden mozzanata és fontos részlete megfelelően legyen dokumentálva. Az első és egyben talán a legfontosabb ez a tanulmány, ami a legtöbb információt és tanulságot hordozza magában. Összefoglalja a kutatás folyamatát az elejétől a végéig és minden lényeges információt tartalmaz, ami a céljaink és ambícióink megértéséhez szükségesek. A második a laboratóriumunk weboldala (link), ahol igyekszünk összegyűjteni a jövő hallgatói számára az igazán lényeges információkat, amik a projektjeink során készült programok futtatásához és megértéséhez elengedhetetlenek. A harmadik a kódban elhelyezett kommentjeim, amiket igyekeztem minél inkább minimalizálni és egyértelművé tenni. Olyan programozóként, akinek a szakmai tapasztalatának 90 százaléka objektum orientált környezetben keletkezett, jól elkülöníthető blokkokat definiáltam a fejlesztéseim során, így az a kevés komment is egy jól átlátható dokumentációként szolgálhat a fejlesztők számára. Végül az utolsó platform a GitHub (link), ahol azok számára vezettem dokumentációt a fejlesztéseim során, hogy ne csak a mi egyetemünk hallgatói számára szolgáljak kutatási eredményekkel, hanem a világ bármely embere számára is, aki érdeklődik a téma iránt. Amellett, hogy itt megtalálható az a leírás, ami alapján a kódom beüzemelhető és futtatható, végig követhető a projektem forráskódjainak fejlődési szakasza, ami további információval szolgálhat, a kutatásom során alkalmazott eszmefuttatásaimmal kapcsolatban.

3. Előzmények

A témában fellelhető szakirodalmi alkotások mennyisége sajnálatos módon igen csekély. Lévén, hogy egy olyan technológiáról van szó, ami csupán pár éve törte be magát a köztudatba, még nem alakult ki az a megbízható forrásokkal szolgáló környezet, mint ami egy évtizedek óta ismert technológia esetében nagy eséllyel megtalálható.

Az ehhez hasonló fejlesztések esetében leginkább a nyílt forráskódú projektekre és azok leírásaira lehet hagyatkozni, ám ebben az esetben nagyon körültekintően és kritikus szemmel kell szemlélődni, mivel köztudottan ezek a projektek sokszor rejthetnek hibákat. Ezek a hibák pedig egy nagyobb méretű drón esetében akár komoly következményekkel is járhatnak, mint ahogy az a 2017 május 9-én rendezett Golden State Race Series biciklis versenyen is megtörtént, ahol az egyik résztvevő biciklijét találta el egy drón miután belecsapódott egy közeli fába. [1]

3.1. Korábbi kutatások

A kutatást megelőzően már egy félévet foglalkoztam a rendszer felépítésével az önálló laboratórium keretein belül. Itt felkutattam a számomra elérhető drónok jellemzőit és több tesztet is végeztem a kiválasztott AR.Drone 2.0-val. Megismerkedtem a ROS rendszerrel, ami széleskörű megoldásokat tartalmaz a robotikával kapcsolatos kutatásokhoz. Beüzemeltük az egyetem birtokában lévő OptiTrack kamerarendszert és több tesztet is végeztünk vele. Megtanultam rendszer szoftverének a Motive-nak a használatát, a kalibrálás folyamatát, a pontos paraméter beállításokat és a kamerák megfelelő elhelyezését a saját tesztkörnyezetünkben. Sikerült definiálni több különálló objektumot, amiket viszonylag hatékonyan el tud szeparálni a szoftver, ezzel elősegítve több projekt egyidejű fejlesztését és tesztelését.

A projekt eredményeként sikerült létrehozni egy olyan teszt környezetet, ahol a ROS segítségével képes volt a rendszer egy helyen kezelni a drón és az OptiTrack adatait. Ehhez viszont elengedhetetlen a ROS beható ismerete, mivel innentől már minden fejlesztés ott történik. Meg kellett tanulnom ROS csomagot készíteni, fejleszteni és használni. Megismerkedtem a csomagok közti kommunikációs csatornákkal és az egyes programok paramétereit tároló szerver működésével. Ez mind lényeges részét képezték a további kutatások elvégzéséhez, mivel itt már magasabb szintű problémákkal kellett foglalkozni, így nem lehetett olyan rész, ami nem világos a számomra.

3.2. QR kód felismerésére alkalmas megoldás keresése

Mielőtt bele fogtam volna a komolyabb fejlesztésekbe, megoldást kellett találnom arra a problémára, hogy a drón kamerájának segítségével ismerjek fel QR kódokat. Felkutattam a ROS rendszerhez írt, különböző megoldásokkal rendelkező csomagokat, amik célzottan ilyen feladatok ellátására alkalmasak. Három különböző csomag jött szóba, amiket egyenként megvizsgáltam, hogy melyik lenne számomra a leginkább alkalmas.

Az első a zbar_ros nevű csomag volt. Aki valaha foglalkozott QR vagy vonalkód detektálással és dekódolással az valószínűleg találkozott már a ZBar nevű nyílt forráskódú programmal. Sok nyelven megtalálható az implementációja és a széleskörű támogatottság eredményeként képes nagyon gyorsan detektálni egy képi kódot. Nagy hátránya viszont, hogy állóképekre van optimalizálva, és néhány további tervezési tulajdonság miatt, mint például, hogy képtelen egyszerre több kódot detektálni önállóan, alkalmatlannak bizonyult a projekt számára.

A második a visp_auto_tracker. Ez a csomag már több lehetőséggel rendelkezik. Képes a képen megkeresni egy QR kódot, majd lekövetni azt és figyelnie a pozícióját és az orientációját. Ez már több kód egyidejű követésére is alkalmas ellentétben a ZBar-ral. Ez a csomag a Visual Servoing Platform része, ami tartalmaz még sok más kapcsolódó programot is. Ilyen például a visp_camera_calibration ami egyedi minták alapján segíti a kamera kalibrációját a minél hatékonyabb felismerés érdekében, illetve a visp_tracker ami már nem a QR kódok hanem háromdimenziós objektumok felismerésére és követésére képes, ami igazán hasznos lehet további kutatások esetén. Bár ez a csomag már képes a legtöbb feladat elvégzésére, amit a projekt sikeressége igényel, de még mindig nem elég megbízható a mozgás közbeni kamera képéről való detektálásra.

Egy fórum bejegyzés alapján találtam rá a visp_auto_tracker egyik versenytársára az ar_track_alvar nevű kódfelismerő csomagra, ami a harmadik és egyben az utolsó megvizsgált program a listában. A bejegyzés szerint ilyen instabil kép esetén, ami az AR drónnál előfordul, sokkal hatékonyabb ez a csomag, mint az előző. Ez is hasonló képességekkel rendelkezik de, az online vélemények szerint a különböző szűrő algoritmusok itt hatékonyabban járnak el a rossz minőségű videókép feldolgozása esetén.

3.3. Hasonló alkotások

Természetesen mielőtt belevágtam volna a programozásba, utánanéztem, hogy milyen hasonló projektek és megoldások léteznek, amikből ötleteket meríthetek. Hasznos lehet más megközelítéseket is számba venni, nem csak azt, ami először az eszünkbe jut. Előfordulhat, hogy egy-egy problémára mások sokkal robusztusabb megoldásokat is találtak már.

Rövid keresés után rátaláltam egy kísértetiesen hasonló projektre (link), ami szerintem egy hasonló kutatás eredményként jöhetett létre. A projekt neve ar2landing_neural, ahol a szerző nem valódi hardverekkel, hanem szimulált környezetben végezte a fejlesztéseket. Érdekes felvetésekkel találkoztam ebben a projektben, mert itt egy neurális háló segítségével hajtott végre landolást a drón, egy QR kód tetején. Sajnálatos módon ez a projekt már 2 éve nem élvezi a fejlesztője figyelmét, így a mai ROS rendszerrel már sok inkompatibilitást mutat. Sok munka lett volna életet lehelni bele, tehát úgy döntöttem, maradok a forráskódok átolvasásánál, hátha szolgálhatnak valamilyen plusz információval.

A kód sokat elmondott a szerzője gondolatmenetéről, amiből aztán később én is hasznosítottam pár gondolatot a saját projektemben. Bár itt nem volt jelen az OptiTrack rendszer, mégis sok segítséget nyújtott a drón és az ar_track_alvar összehangolásában. Sajnos a többi alkalmazott megoldás nekem a valóságban nem bizonyult elég megbízhatónak, mivel a szimuláció során sok fizikai tényező nem érvényesült, ami a valós tesztelésnél teljesen megváltoztatta a drón viselkedését.

4. A tervezés részletes leírása

Miután sikeresen teszteltem a drón, az OptiTrack kamerarendszer a Robot Operating System és a QR kód detektálás működését, kezdődhetett a fejlesztői munka. Ez a fejezet részleteibe menően bemutatja a fejlesztés során fellépő problémákat és azok megoldásait.

4.1. A munka során felhasznált eszközök

A kutatás elvégzéséhez sok hardveres és szoftveres eszköz nyújtott segítséget. Ebben az alfejezetben ezeket szeretném ismertetni és a fontosabb jellemzőiket bemutatni. Fontos megjegyeznem, hogy a fejlesztés során kialakult rendszer nem indokolja, hogy pontosan ezeket az eszközöket használja fel valaki hasonló munkája során. Ez csak egy olyan irányvonalat képvisel, ami megbízható hátteret biztosít a kutatói munka támogatásához.

4.1.1. OptiTrack kamerák

Korábban már szerepelt a motion capture technológia jelentősége és gyakorlati haszna. A hardver, ami mindezt biztosítja számunkra, az OptiTrack cég által forgalmazott kamerarendszer. Sok féle kamerát tartalmaz a repertoárjuk, de nem mindegyik alkalmas arra a jellegű munkára, amit ezalatt a kutatás alatt végeztem. Ezek a speciális kamerák, az infravörös fényre érzékeny lencsékkel és szűrőkkel vannak ellátva, valamint a lencséjük körül egy gyűrűvel, ami infravörös fény kibocsátására alkalmas LED diódákat tartalmaz. Továbbá fel vannak még szerelve egy állapotjelző LEDeket tartalmazó másik gyűrűvel is a lencséjük körül, ami segíti a kalibrálás menetét.

Az egyetem 8 darab Prime 13 típusú kamerával rendelkezik, amik kiválóan használhatók olyan méretű terek lefedésére, mint például az egyetem edzőterme, ami végül a tesztek helyszínéül is szolgált. Ezek a kamerák egyenként 13 megapixelesek, 1280 x 1024 pixel arányokkal. Képesek 240 képkocka per másodperc sebességgel képet rögzíteni, horizontálisan 56 míg vertikálisan 46 fokos betekintési szög mellett. Az átlagos késleltetésük 4,2 ms ami igazán gyorsnak mondható és ez cseppet sem von le a kutatás minőségéből. A 8 kamerára a csekély betekintési szögek miatt volt szükség, mivel legalább ennyi kamera kell egy nagyjából 40 négyzetméteres terület lefedéséhez.

Ezek a kamerák ethernet kábelen keresztül kommunikálnak az őket vezérlő számítógéppel és ugyanezen a kábelen tudják felvenni a működésükhöz szükséges áramot. Ebből kifolyólag olyan switch szükséges hozzájuk, ami képes a Power over Ethernet funkcióra.

4.1. ábra. Prime 13 típusú kamera

4.1.2. Markerek

A motion capture technológiához, speciális markerekre is szükség van, ezek is elengedhetetlen részei a rendszernek. Ezek a markerek lényegében olyan golyók, amiknek a felülete magas intenzitással képes visszaverni a fényt. Ezek közül is az infra tartományba tartozó, magas hullámhosszú fényt tudja a leghatékonyabban reflektálni. Így a kamerák, jól el tudják különíteni őket a környezetüktől.

4.2. ábra. Fényvisszaverő marker és egy hozzá tartozó rögzítő talp

4.1.3. Motive

Az OptiTrack rendszer hivatalos szoftvere a Motive. Képes a kamerák által modellezett háromdimenziós tér megjelenítésére és annak konfigurálására. [9] Ennek segítéségével végrehajthatunk különböző koordináta transzformációkat, definiálhatunk objektumokat, amik több markert csoportosítanak össze és közvetíthetjük a detektált markerek pozícióit, illetve az objektumok orientáció adatait.

A kamerákat UTP kábelekkel csatlakoztathatjuk egy switchen keresztül a Motive szoftvert futtató géphez, ami előzetes beállítások nélkül azonnal felismeri őket. A kamerák kalibrálásához néhány ezer mintát kell gyűjteniük a megfigyelt térből. A mintákat egy erre a célra készített kalibráló rúddal generálhatjuk manuálisan. [10]

4.3. ábra. Kalibráló rúd, 3 markerrel felszerelve

A kalibrálást követően a szoftver kiszámítja a kamerák egymáshoz viszonyított pozícióit és elkészül a modellezett tér, amiben az eszközök helyzetét valós időben figyelhetjük meg.

4.1.4. Robot Operating System

A Robot Operating System vagy rövidebben csak ROS egy nyílt forráskódú, általánosított „operációs rendszer” robotokhoz. Megvalósítja a hardver szintű absztrakciót, képes kezelni a hardverek vezérlését és az egyes folyamatok közti kommunikációt.

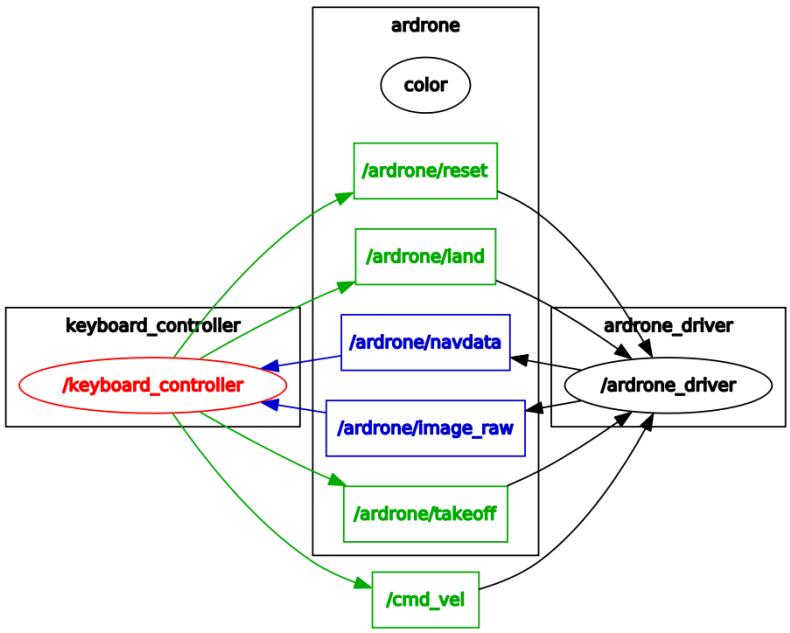
A ROS rengeteg olyan eszközt és könyvtárat biztosít, ami lehetővé teszi a különböző platformokon való fejlesztést. Támogatja a kód újra felhasználást, ami nagymértékben megkönnyíti a fejlesztők munkáját. Rendelkezik beépített vizuális megjelenítővel is, ami kiváló segítséget nyújt a teszt környezetünk szemléltetéséhez a kódot futtató gépen.

A kutatás során a Kinetic verziójú ROS-t használtam, ami teljes mértékben kompatibilis a Xenial verziójú Ubuntu rendszerrel és a kapcsolódó alprogramok is mind, teljes körűen támogatják.

A ROS rendszer építőelemei:

- **Node**: futtatható állomány, ami a ROS-t használja a többi node-dal való kommunikációhoz
- **Message**: primitív típusú adat vagy tömb, amit a node-ok küldhetnek és fogadhatnak
- **Topic**: a node-ok üzeneteket publikálhatnak bele, míg mások feliratkozhatnak rá, hogy megkapják az üzeneteket. Lényegében ezekből épül fel a binárisok közti kommunikációs csatorna
- **Service**: egy node-ban implementált szolgáltatás, ami a ROS-on keresztül meghívható
- **Master**: a ROS rendszer központi eleme, ami számon tartja a különböző node-okat, topic-okat, feliratkozásokat és publikálásokat
- **Parameter Server**: központi tároló, ami a node-ok paramétereit tárolja és közvetíti

[3]



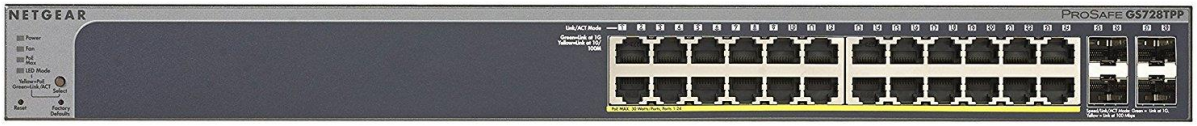
4.4. ábra. Példa a ROS node-ok közötti kommunikáció felépítésre

4.1.5. Számítógépek

A rendszer működtetéséhez 2 darab számítógépre volt szükség. Az egyik Windows 10 operációs rendszerrel, míg a másik VirtualBox-on keresztül Ubuntu Xenial operációs rendszerrel felszerelve. A Windows-os gépen található a Motive ami közvetíti a hálózaton keresztül az objektumok adatait. A másik gép a Robot Operating System és a hozzá kapcsolódó programok futtatásáért felelős. Itt összpontosul a kamerarendszerből kapott és a drón által szolgáltatott összes adat és az itt futtatott programokon keresztül lehetséges a parancsok közvetítése a drón rendszeréhez.

4.1.6. Switch

Az OptiTrack kamerái nem rendelkeznek külön tápellátással. A működésükhöz szükséges áramot a kommunikációhoz is használt ethernet porton keresztül veszik fel. Ezt a funkciót egy átlagos switch nem képes ellátni, mivel nem képes akkor feszültség felvételére, ami ehhez szükséges. Ebből kifolyólag egy olyan switchre volt szükség, ami képes egyszerre akár 8 kamera árammal való ellátására is. A kutatás során használt switch a NETGEAR ProSafe GS728TPP. Ez a switch 24 darab gigabit ethernet porttal rendelkezik, amelyekből mindegyik képes a Power over Ethernet funkció ellátására. Összesen 384 watt leadására képes, ami bőségesen elég a 8 darab Prime 13 típusú kamera működtetéséhez. Továbbá ez a switch alkalmas lesz akkor is, ha az egyetem több kamerával bővítené a tárházát.



4.5. ábra. NETGEAR ProSafe GS728TPP típusú switch patch panele

4.1.7. Router

Rendelkezésre állt még egy LINKSYS WRT1900AC típusú router is, ami képes automatikusan csatlakozni a drón által közvetített Wi-Fi hálózatra. Ez azért lehet fontos, mert a drón fedélzeti számítógépén található rendszer csak olyan hálózathoz tud csatlakozni, amik nincsenek ellátva titkosítással. Így megoldható, hogy a vezérlést végző laptop ne legyen kábelekhez kötve. Ez a router alkalmas akár a legújabb AC szabványú Wi-Fi hálózat létrehozására is, amit tanácsos használni mivel ez az 5 GHZ-es tartományban kommunikál az eszközökkel nem pedig a régi routerek által használt 2,4 GHZ-es tartományban, ami napjainkban kezd kissé túl zsúfolt lenni. Ráadásul ezzel a szabvánnyal sokkal gyorsabb kommunikációt is elérhetünk.

4.6. ábra. Linksys WRT1900AC típusú router és a rajta látható állapotjelző sáv

4.1.8. Drón

A kutatás során a Parrot cég által gyártott AR.Drone 2.0 volt a segítségemre, ami egy távolról vezérelhető, kifejezetten beltéri használatra felkészített négyrotoros pilóta nélküli repülőgép. Wi-Fi kapcsolaton keresztül képes kommunikálni más eszközökkel. Tartalmaz egy 4GB-os tárhelyet ami lehetővé teszi, hogy egyedi algoritmusokat töltsünk fel és automatizáljuk a működését. A drón alsó részén található egy ultrahangos távolságmérő és egy légnyomásmérő szenzor, továbbá be van építve még egy 3 tengelyű giroszkóp, egy 3 tengelyű gyorsulásmérő szenzor és egy magnetométer, amik a drón stabilan tartását hivatottak biztosítani. [2]

4.7. ábra. A Parrot cég AR.Drone 2.0 típusú drónja

4.2. A tesztkörnyezet felépítése

Ez az alfejezetben a teszteléshez használt környezet felépítését és kalibrálásának folyamatát szeretné bemutatni az olvasó számára. Itt is megjegyezném, hogy ezek nem szigorú, inkább csak ajánlott feltételei a helyes használatnak.

4.2.1. Tesztelési helyiségek

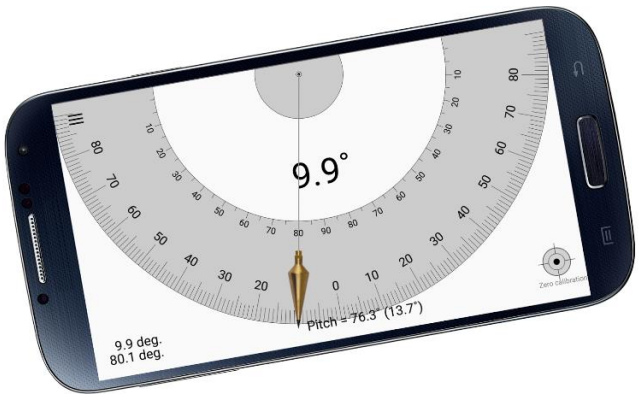
Alapvetően két különböző helyszínen történt a kutatás közbeni tesztelés. Az egyik a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának oktatói klubja, míg a másik ugyanezen Kar edzőterme volt.

Az oktatói klub alapterülete nem haladja meg a 15-20 négyzetmétert, amiből további lényeges területet foglalnak el az oda tartozó bútorok. Így mondhatom, hogy csekély hely maradt a drónnal való repülésekhez. A belmagasság nem okozott különösebb problémát, viszont a drón közelében lévő tárgyak erősen befolyásolták a repülés trajektóriáját. Ez főleg Bernoulli törvényéből adódik, ami azt mondja ki, hogy egy közeg áramlásakor a sebesség növelése a nyomás csökkenésével jár. Vegyük például azt az esetet mikor a drón közel repül egy falhoz. Ilyenkor a drón mindkét oldalán ugyanakkora sebességgel forognak a rotorok. A légnyomás mindkét oldalon alacsonyabb lesz az eredetihez képest. Azon az oldalon, ahol nincs fal, van elegendő levegő, ami a csökkent nyomású területre áramlik. Ellenben a másik oldallal, ahol a fal miatt nem jut elegendő levegő a területre, így az a drón másik oldala felől áramlik át, ami által a fal mintegy magához szívja a drónt. A jelenség megtekinthető a mellékletben található sucked_by_object.mp4 felvételen.

Néhány alkalommal lehetőségem adódott az edzőteremben való tesztelésre is. Itt már nem jelentkeztek az előző helyiségnél említett problémák és szinte tökéletesen tudta tartani a drón a kívánt pályát repülés közben. A hatalmas belmagasság és a közel 40 négyzetméteres, tesztelésre kijelölt terület nem szabott több határt a repülések közben.

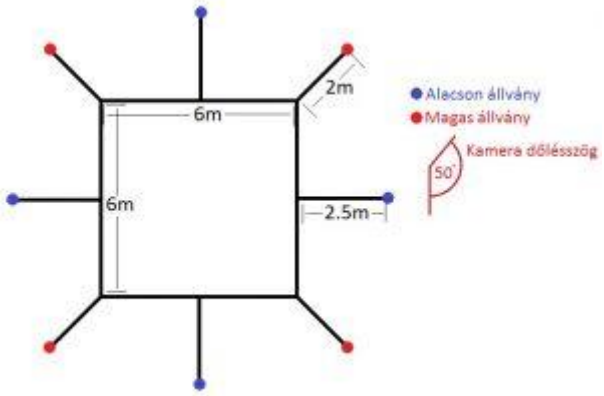
4.2.2. Kamerák elhelyezése

A kamerák, speciálisan erre a célra készített állványokon helyezkedtek el. Ezek az állványok nagy súllyal rendelkeznek, hogy a drón ne tudja őket belengetni repülés közben, ami elrontaná a kamerarendszer kalibrációját. Négy darab alacsonyabb és négy darab magasabb állvány állt rendelkezésre. A magasabb állványok 3 méteres magasságra voltak kiengedve, míg az alacsonyabbak körülbelül 1,5 méteres magasságra. Az alacsonyabb állványokon elhelyezkedő kamerák vízszintesen előre néztek, míg a magasabb állványokon lévők 50 fokos szöget zártak közre a vízszintessel. A dőlésszög lemérését a Protractor nevű Android alkalmazás segítette.



4.8. ábra. A Protractor nevű Android alkalmazás

Az állványok elhelyezése egy négyzet alapú területen történt, az alábbi képen látható módon.



4.9. ábra. Kameraállványok elhelyezésének alaprajza

4.2.3. Hálózati beállítások

A kamerák a már korábban is említett switchbe voltak közvetlen csatlakoztatva. A switchez csatlakozott még a Windows-os számítógép és a router is. A számítógép fogadta a kamerák képeit, majd közvetítette az objektumok kiszámolt adatait a switchen keresztül a hálózatra. A router létrehozott egy vezeték nélküli alhálózatot, amihez csatlakozhatott a másik számítógép és lehetőség szerint a drón is. A routeren keresztül az internetet is be lehet vezetni a hálózatba, így a tesztelés közben nem kell átváltani másik Wi-Fi-re.

Mivel az Ubuntu rendszert egy virtuális gépen keresztül futtattam, akadtak nehézségek a hálózati kommunikációkkal. Sajnálatos módon a drón hiába csatlakozott a router-hez, azt a virtuális gépről nem értem el. Kipróbáltam több féle konfigurációt is, végül csak egy olyan akadt, amivel minden működött. A gépet etherneten keresztül a switchbe csatlakoztatva és a géppel a drón Wi-Fi hálózatához csatlakozva. A VirtualBox a gép ethernet portját bridgelve, míg a vezeték nélküli hálózatot NAT-olva érte el. Így lehetőség adódott mindegyik eszközzel való kommunikációra, bár a gép így kábelhez kötve maradt.

4.2.4. Drón konfigurálása

A drón konfigurálható, hogy csatlakozzon egy titkosítatlan vezeték nélküli hálózathoz. Ehhez először fel kell csatlakozni a drón Wi-Fi-jére, majd telnet segítségével bejelentkezni a fedélzeti rendszerébe. Létre kell hozni rajta egy shell scriptet, ami a következőket tartalmazza:

```
killall udhcpd
ifconfig ath0 down
iwconfig ath0 mode managed essid [SSID]
ifconfig ath0 [DESIRED IP] netmask 255.255.255.0 up
route add default gw [GATEWAY IP]
```

A szkriptben látható kék részek hálózat függőek, ezért azokat előbb ki kell deríteni az adott hálózatnál, majd behelyettesíteni őket. A fájl elmentése után engedélyezni kell a futtatását, amit a `chmod +x [FILE PATH]` paranccsal tehetünk meg. Végül zárjuk a telnet kapcsolatot.

Ezek után a legközelebbi használat esetén csak fel kell csatlakozni a drón hálózatára és lefuttatni ezt a parancsot: `echo "./[FILE PATH]" | telnet [DRONE IP]`

4.2.5. Akadálypálya felépítése

Az akadálypályára több terv is volt kezdetben, amik végül nem bizonyultak megvalósításra érdemesnek vagy logikusnak. Végül egy olyan akadálypálya jött létre, amit egyszerű összeállítani, mégis sok különböző feladat tesztelésére alkalmas lehet. Összesen csak a kalibráló rúd felső része, valamint a drón dobozára volt szükség. Ezen kívül a QR kódokat kellett kinyomtatni a pálya véglegesítéséhez. Praktikus ez a megoldás, mert nem kell plusz eszköz a felállításához. A kalibráló rúd három markerét egyesítettem egy cél objektummá, aminek a jelentősége, hogy ezen objektum fölé kellett berepülnie a drónnak a saját pozíciója ismeretében. Innen kellett detektálnia a dobozon elhelyezett QR kódot, majd a kód alapján egy bizonyos feladatot végrehajtania. A projekt végén két ilyen feladat valósult meg. Az egyik a doboz felett való átrepülés, majd landolás, a másik a doboz mellett való elrepülés, majd landolás a doboz mögött. Mindkét feladat sikeres tesztje megtekinthető a mellékletben található videó felvételeken.

4.10. ábra. A drón doboza a ráragasztott QR kóddal

4.3. A munka során felhasznált forráskódok

A kutatás folyamán sok olyan nyílt forráskódú programot használtam fel, amik segítik lekezelni az alacsony szintű, hardverközeli programozást és interfészként szolgálnak számomra, hogy magasabb szinten tudjam lekezelni a problémákat. Ezek mind a Robot Operating System részeként működnek együtt és legtöbbjük folyamatos fejlesztés alatt áll, így fontos volt ez esetben is a körültekintő hozzáállás a fejlesztés folyamán.

4.3.1. ardrone_autonomy

Az ardrone_autonomy egy ROS driver a Parrot AR.Drone 1.0 és 2.0 típusú drónokhoz. A ROS-on belül egy külön node-ként jelenik meg, ami a következő fontos topic-okat közvetíti:

- **/ardrone/navdata**: a drón navigációs adatait közvetíti
- **/ardrone/image_raw**: a kamerák képeit közvetíti

Továbbá még számos *service* is meghívható benne, amivel a drón fedélzeti számítógépét vezérelhetjük és előre megírt animációkat hívhatunk elő. [4]

4.3.2. ardrone_tutorials

Az ardrone_autonomy fölé épült absztrakciós szint, ami segít a drón vezérlését megkönnyíteni. Sajnálatos módon akadtak a forráskódjának olyan részei, amik nem átgondolt tervezésre utalnak. A továbbiakban igyekszem javasolni néhány változtatást a fejlesztői felé. Ezekből a tervezési hibákból adódóan úgy kellett néhány dolgot lekezelnem a saját programomban, amik szembe mennek a saját fejlesztői elveimmel és így bár kis mértékben de instabilitást okozhatnak. Ugyanakkor érdemes felhasználni, mert a későbbi javítások esetén a saját kódunkon nem vagy csak kis mértékben kell változtatni.

4.3.3. ar_track_alvar

Mint korábban már említettem ez a kis program képes a megadott kameraképen detektálni és dekódolni a QR kódokat. Helyette van lehetőség más megoldások alkalmazására, de a fejlesztés során ez bizonyult a leginkább alkalmasnak.

4.3.4. Virtual-Reality Peripheral Network

A VRPN egy platform független rendszer, ami interfészként működik szoftverek és fizikai eszközök között. Képes kezelni a ROS-sal való topic orientált kommunikációt. Szerver – Kliens modell alapján működik és Internet Protokoll segítségével kommunikál más eszközökkel. [5] A Motive is ezen a rendszeren keresztül képes publikálni az objektumok mért adatait, amit aztán a ROS dolgoz fel és végül lekérdezhető a saját forráskódunkban is. Ehhez szükség van a vrpn_client_ros nevű csomag feltelepítésére.

Sajnos ezen a rendszeren keresztül nem képes a Motive közvetíteni a gyorsulás irányára vonatkozó adatokat. Ennek okára nem sikerült logikus magyarázatot találni sem az interneten, sem pedig az Optitrack fórumain. Amennyiben ez a probléma megoldódik a későbbiekben, ezekkel az adatokkal is tovább lehet fejleszteni a szkript döntési algoritmusát.

4.4. Saját ROS csomag

A fejlesztői munka során egy saját ROS csomagot hoztam létre, ami minden szükséges összetevőt tartalmaz a program futtatásához. Törekedtem arra, hogy a letöltése után szinte azonnal futtatható legyen, különösebb konfigurálások nélkül is. Az egész ROS csomag megtalálható a mellékletek között.

4.4.1. Nyelvezete

A forráskódot python nyelven írtam, amit teljeskörűen támogat a ROS. Annyi nehézség adódott, hogy ezelőtt még soha nem programoztam ezen a nyelven, de még csak másik szkript nyelven sem. Fogalmam sem volt mit tartogat ez a világ egy olyan ember számára, aki az objektum orientált világ ölelésében nevelkedett. Ennek ellenére azért választottam ezt a nyelvet, mert tisztában voltam vele, hogy a futtatásához elegendő egy jól működő interpreter és nem kell a fordítások és linkelések adta problémák miatt bosszúskodni.

A fejlesztések alatt végig igyekeztem tartani magam a PEP8 által meghatározott szigorú szabályokhoz. Ez nagyrészt sikerült is egy-két kivétel mellett. Fontosnak tartottam, hogy jól olvasható, könnyen megérthető kódot adjak ki a kezeim közül, így a szabályok betartása mellett, igyekeztem minél érthetőbb kommentelést is vezetni a kódban. Így nem kell elvesznie az olvasónak a részletekben.

4.4.2. Felépítése

Örömömre szolgált az a felismerés, hogy szkriptnyelv lévén a python igazán barátságos megoldásokat kínál az objektum orientált szemléletű embereknek is. Sikerült számomra jól átlátható, logikailag elkülöníthető blokkokba szerveznem a kódomat. Az egész forráskód lényegében három logikai egységre lett tagolva, plusz egy úgynevezett launch fájlra, ami egy ROS program lényegi részeinek elindításáért felelős.

Az első és legfontosabb logikai egység egy központi rész, ahol összefut minden szál. Ezt én operator-nak neveztem el, mivel ez felelős a többi egység irányításáért. Itt definiálódik a drón controller objektuma és a másik két egység is. Mindegyik egység megkapja a drón irányításáért felelős objektumot, miután az operator elindította a drónt.

A második egység, az OptitrackController osztály. A elnevezés onnan ered, hogy ez az objektum felelős a Motive által szolgáltatott adatok alapján való tájékozódásért és manőverezésért. Az inicializáló függvényében megkapja a drón irányításához szükséges objektumot, definiálja a repülés során használt változókat és feliratkozik minden, számára fontos topic-ra, amik a következők:

- **/vrpn_client_node/ardrone/pose**
  A drón pozíció és orientációs adatai egyenesen a Motive VRPN szerverétől
- **/vrpn_client_node/destination/pose**
  A kalibráló rúd pozíció adatai szintén a Motive VRPN szerverétől
- **/ardrone/pose**
  A drón pozíció adatainak transzformált változata. Ezek megegyeznek a drón által értelmezett koordinációs rendszer béli adatokkal.
- **/visualization_marker**
  A detektált QR kódok drónhoz viszonyított, relatív pozíció és orientációs adatai az ar_track_alvar rendszeréből. Itt csak akkor érkezik üzenet, ha sikerült a detektálás.

A harmadik egység…

4.4.3. Verziókövetés

5. Eredmények

A feladatunk végére érve kialakult a végső fejlesztői környezet, ami mindent biztosít ahhoz, hogy a fejlesztő által implementált kódot probléma nélkül tudja futtatni.

Az *ardrone_tutorials* nevű, nyílt forráskódú projekt tartalmazza a teszteléshez szükséges vezérlési parancsokat és a mintájára könnyedén készíthető saját alkalmazás. A projekt forráskódja python nyelven íródott, és az *ardrone_autonomy* által biztosított parancsokat használja.

A topic-okban közvetített adatokat könnyedén vizualizálhatjuk *rviz*-zel a ROS beépített grafikus megjelenítőjével. A kezelőfelületén megjelennek az aktuális topic-ok és a benne szereplő adatok, amelyeket megjeleníthetünk 2D-s vagy 3D-s környezetben. [12]

Amennyiben nem labor környezetben vagyunk és nincs lehetőségünk az AR.Drone-nal és OptiTrack kamerákkal tesztelni, használhatjuk a *tum_simulator* nevű programot, ami részben képes kiváltani a valós rendszert. Ez az otthoni fejlesztést is lehetővé teszi. [13]

6. Értékelés

Kritika

továbbfejlesztés (pl: swarming, mozgó objektumra landolás)

szimulátor

7. Összefoglalás

teljesítettségi mutató a kiírás függvényében

8. Köszönetnyilvánítás

Szeretnék köszönetet mondani a karnak a biztosított eszközökért és támogatásért. Nagyra tartom, hogy ilyen kifinomult dolgokkal van lehetőségünk dolgozni.

Továbbá köszönet illeti a már korábban említett nyílt forráskódú szoftver csomagok és könyvtárak fejlesztőit, akik remek munkát végeztek a témában.

9. Irodalomjegyzék

[1] „Golden State Race Series,” [Online].

[2] „Parrot AR.DRONE 2.0 Power Edition | Parrot Store Official,” Parrot, 2017. [Online]. Available: https://www.parrot.com/us/drones/parrot-ardrone-20-power-%C3%A9dition. [Hozzáférés dátuma: 04 05 2017].

[3] „Documentation - ROS Wiki,” 31 03 2017. [Online]. Available: http://wiki.ros.org/. [Hozzáférés dátuma: 05 05 2017].

[4] „ardrone_autonomy — ardrone_autonomy indigo-devel documentation,” 04 2014. [Online]. Available: http://ardrone-autonomy.readthedocs.io/en/latest/. [Hozzáférés dátuma: 05 05 2017].

[5] „vrpn Wiki,” 21 01 2017. [Online]. Available: https://github.com/vrpn/vrpn/wiki. [Hozzáférés dátuma: 06 05 2017].

[6] „Introduction · PX4 Developer Guide,” 2017. [Online]. Available: https://dev.px4.io/en/. [Hozzáférés dátuma: 07 05 2017].

[7] R. Roy és V. Bommakanti, „ODROID-XU4 Beginner's Guide,” 2015. [Online]. Available: https://magazine.odroid.com/wp-content/uploads/odroid-xu4-user-manual.pdf.

[8] A. Bartha és B. Petrovicz, „uavLab,” 2017. [Online]. Available: https://uavlab.itk.ppke.hu/. [Hozzáférés dátuma: 08 05 2017].

[9] „Motive Documentation - NaturalPoint Product Documentation,” 19 08 2016. [Online]. Available: http://wiki.optitrack.com/index.php?title=Motive_Documentation. [Hozzáférés dátuma: 06 05 2017].

[10] „OptiTrack - Calibration Tools,” 2017. [Online]. Available: https://optitrack.com/products/tools/. [Hozzáférés dátuma: 06 05 2017].

[11] „Rigid Body Tracking - NaturalPoint Product Documentation,” NaturalPoint Corporation, 25 01 2017. [Online]. Available: http://wiki.optitrack.com/index.php?title=Rigid_Body_Tracking. [Hozzáférés dátuma: 07 05 2017].

[12] „rviz - ROS Wiki,” 15 06 2016. [Online]. Available: http://wiki.ros.org/rviz. [Hozzáférés dátuma: 07 05 2017].

[13] „tum_simulator - ROS Wiki,” 12 05 2014. [Online]. Available: http://wiki.ros.org/tum_simulator. [Hozzáférés dátuma: 07 05 2017].

[14] „Press Releases | Airobotics,” Airobotics, 2017. [Online]. Available: http://www.airobotics.co.il/press-releases/. [Hozzáférés dátuma: 08 05 2017].

[15] „How Drone Will be Used In The Future - Business Insider,” 16 12 2013. [Online]. Available: http://www.businessinsider.com/how-drones-will-be-used-in-the-future-2013-12. [Hozzáférés dátuma: 08 05 2017].

[16] „From Driverless Cars to Automated Drones - The Future is Already Here | Airobotics,” 24 11 2016. [Online]. Available: http://www.airobotics.co.il/blog/driverless-cars-automated-drones-future-already/. [Hozzáférés dátuma: 08 05 2017].

10. Mellékletek

- forráskód
- videó