VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAVIGACE A ŘÍZENÍ KVADROKOPTÉRY

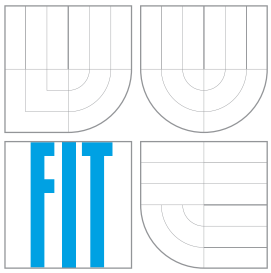
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

KAREL DOLEŽAL

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAVIGACE A ŘÍZENÍ KVADROKOPTÉRY

QUADROCOPTER NAVIGATION AND CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — KAREL DOLEŽAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID HERMAN
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2011/2012

Zadání bakalářské práce

Řešitel: **Doležal Karel**
Obor: Informační technologie
Téma: **Navigace a řízení kvadrokoptéry**
Quadrocopter Navigation and Control
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se s kvadrokoptérou AR Drone a se sadou dodávaných programových nástrojů. Senzorický systém kvadrokoptéry rozšiřte o GPS senzor pro účely autonomní navigace ve venkovním prostředí.
2. Navrhněte navigační architekturu pro autonomní přesun kvadrokoptéry do cílové pozice zadané GPS koordináty včetně autonomního přistání na přistávací plošinu. Umožněte zadat více podcílů a definovat tak přibližnou trajektorii letu.
3. Navrženou navigační architekturu implementujte a otestuje v reálném prostředí na platformě AR Drone.
4. Zhodnoťte dosažené výsledky a diskutujte možnosti dalšího rozšíření.

Literatura:

- Thrun S.: Probabilistic Robotics, MIT Press, 2005, ISBN 026220162
- *AR Drone* [online]. 2011 [cit. 2011-11-01]. Dostupné z WWW: <http://ardrone.parrot.com/parrot-ar-drone/en/>.
- Dle specifikace vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herman David, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2011
Datum odevzdání: 16. května 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Práce se zabývá autonomní navigací kvadrokoptéry ve venkovním prostředí na platformě AR.Drone. Úkolem je samostatný přesun po zadané trase a autonomní přistání na plošině umístěné v cílovém bodě. Práce popisuje platformu AR.Drone včetně dostupných vývojových nástrojů a postup rozšíření senzorů kvadrokoptéry o GPS a magnetický kompas. Dále je představena navržená navigační architektura, jsou popsány role jejích klíčových částí a řízení v jednotlivých fázích letu. Přistání v cíli je řízeno podle kamer umístěných na kvadrokoptéře s vyhledáváním na základě výrazné barvy. Práce se také zabývá detekcí překážek z přijímaného videa pomocí výpočtu optického toku, potlačováním pohybů kvadrokoptéry a vyhýbáním se významným změnám v obraze. Řídicí program je implementován včetně pomocné aplikace pro ladění ze záznamu a úspěšně otestován v reálném prostředí.

Abstract

This paper focuses on autonomous navigation of AR.Drone quadrocopter in outdoor environment. The goal is to follow a specified route and land autonomously on a platform placed at the destination. Firstly, the AR.Drone platform, its development kit and sensor extension with GPS and a magnetic compass are described. Then, the navigation architecture of a control program is presented describing important blocks and its' individual tactics. Localization of the landing platform is based on its color. The video is also used to detect nearby obstacles using optical flow calculation suppressing the quadrocopter movements and to avoid the greater changes in the image. The control program implementation is then tested in real environment and the results are presented.

Klíčová slova

kvadrokoptéra, AR.Drone, GPS, optický tok, autonomní navigace, detekce překážek

Keywords

quadrocopter, AR.Drone, GPS, optical flow, autonomous navigation, obstacle detection

Citace

Karel Doležal: Navigace a řízení kvadrokoptéry, bakalářská práce, Brno, FIT VUT v Brně, 2012

Navigace a řízení kvadrokoptéry

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Davida Hermana, s použitím uvedených zdrojů.

.....
Karel Doležal
6. května 2012

Poděkování

Chtěl bych zde poděkovat mému vedoucímu Ing. Davidu Hermanovi za odbornou pomoc, praktické připomínky, neustálou podporu a trpělivost během vzniku této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Kvadrokoptérám se dnes věnuje značná pozornost, jsou studovány jak v robotice, tak v oblasti řízení a regulace a dalších oborech. Tato zařízení mohou být i velice malá a relativně levná, což je činí široce dostupná. Navíc se skládají z ne příliš mechanicky složitých součástí, kromě čtyř pevně umístěných motorů s vrtulemi obvykle neobsahují žádné složitější pohyblivé části, na rozdíl např. od rotoru klasické helikoptéry. Pohon tvoří čtyři vrtule, které způsobují vztlak s řiditelnou intenzitou a jejichž celková síla směrem vzhůru způsobuje vertikální zrychlení. Zvýšený tah na jedné ze stran pak umožňuje kvadrokoptéru naklonit, a tak způsobit horizontální zrychlení, toho je navíc možné dosáhnout v libovolném směru bez ohledu na její natočení, což je příznivé pro obratnost manévrování. U sofistikovanějších modelů lze rovněž pro další zlepšení manévrovatelnosti měnit náklon listů. Další schopností je otáčení letícího stroje kolem svislé osy. Každé dvě sousední vrtule se otáčejí opačným směrem a působí tak na trup kvadrokoptéry opačnými točivými momenty. Ty jsou za běžného stavu v rovnováze, zvýšením otáček jedné dvojice stejně se otáčejících motorů vzhledem k druhé se naopak dosáhne rotace stroje. U běžných helikoptér se k takovému účelu používá pomocný rotor.

1.1 Cíle práce

Práce se zabývá autonomní navigací kvadrokoptéry, která by ve výsledku měla být schopna přesunu ze startovního bodu po trase definované zeměpisnými souřadnicemi na cílové místo. Na této pozici je pak umístěna přistávací plošina, na kterou by měl stroj autonomně přistát. Zamýšleným senzorem globální pozice je při tom přijímač GPS signálu. Protože dnešní GPS přijímače běžně neumožňují zaměřit pozici s metrovou přesností, je volena plošina snadno detekovatelná v blízkém okolí. Může mít např. podobu kvádru výrazné barvy, kterou lze v okolí snadněji rozpoznat.

V této práci je popisováno řešení tohoto navigačního problému v parkovém, nebo jiném podobném přírodním prostředí (louka, pole), nepředpokládá se nasazení v ulicích ve městě, kde se vyskytuje značné množství překážek a vzniká možnost ohrožení dopravy. Důvodem je rovněž přesnost GPS, která je v této aplikaci klíčová, blízko budov snížená a tedy nedostačující. Předpokládá se také let za bezvětří, nejvýše za mírného větru, vzhledem k malým rozměrům a hmotnosti stroje.

1.2 Přehled kapitol

Následující kapitola (kap. 2) je seznámením s platformou AR.Drone, její konstrukcí a hardwarovým vybavením. Popisuje původní účel produktu, jeho schopnosti co se týče automatické stabilizace a podporu pro herní aplikace. Uvádí také dostupné softwarové prostředky pro řízení kvadrokoptéry, dále rozebírá formát příchozích dat a příkazy řízení včetně jejich realizace v odchozí komunikaci.

Kapitola 3 se zabývá výběrem vhodného senzoru globální pozice a formátu výstupních dat podle standardu NMEA se zaměřením na kódování zde využívaných informací o poloze. Dále řeší specifika zvoleného modulu včetně proprietárních informací posílaných v souladu se zmíněným standardem. Dále jsou rozebrány možnosti připojení tohoto senzoru ke kvadrokoptéře a popsáno zvolené řešení včetně jeho kladů a záporů.

Kapitola 4 popisuje návrh navigační architektury, funkci jejích dílčích částí a toky dat mezi nimi. Rozebírá také fáze letu, podmínky přechodu mezi nimi a v nich navrhované řízení. Nakonec popisuje manévry přistání a vyhýbání se překážkám.

Kapitola 5 o zpracování obrazu se věnuje formátu příchozího videa a jeho konverzi pro zpracování knihovnou OpenCV. Kapitola také popisuje vhodné barevné prostory použité při vyhledávání přistávací plošiny. Dále popisuje samotnou detekci plošiny a také využití obrazové informace za letu k detekci překážek v okolí kvadrokoptéry za pomoci výpočtu optického toku.

Kapitola 6 popisuje návrh tříd. Jsou vysvětleny jejich role, chování a vliv na výpočet řízení. Nakonec je popsán pomocný program prohlížeče záznamů, jakožto prostředku pro vývoj a ladění podle záznamů letu, a navázání řídicího programu na AR.Drone SDK.

Kapitola 7 prezentuje výsledky dosažené v reálném prostředí a vliv nastavení parametrů řídicího programu na chování řízení.

V závěru (kap. 8) jsou diskutována možná budoucí rozšíření a vhodná vylepšení řídicího systému a jsou shrnuty dosažené výsledky. V přílohách je popsán obsah přiloženého CD a postup spuštění řídicího programu s kalibrací joysticku.

Kapitola 2

Platforma AR.Drone

AR.Drone je platforma vyvinutá společností Parrot SA. Jde o kvadrokoptéru velikosti 52,5×51,5 cm o váze 420 g. Konstrukce se skládá z řídicí jednotky připevněné uprostřed kříže z karbonových tyčí, na jejichž koncích jsou umístěné třífázové motory s regulátory. Střed s řídicí jednotkou je obklopen trupem z pěnové hmoty, na jehož vrcholu je prostor pro akumulátor. Ten je kryt odnímatelnou svrchní částí trupu. Pro létání uvnitř budov se dodává i větší vrchní kryt s chrániči kolem vrtulí, jak je vidět na obrázku 2.1.

2.1 Vybavení

Hlavní částí řídicí jednotky je deska tištěných spojů s hlavním procesorem Parrot P6 na bázi ARM (součástka ARM926EJ-S) [5]. Procesor běží na frekvenci 468 MHz a má k dispozici 128 MB RAM. K procesoru jsou připojeny dvě kamery, vertikální kamera pro snímání povrchu a horizontální kamera OmniVision ov7725 [15]. Dále je přítomen rozšiřující konektor s USB, a modul Wi-Fi, přes který kvadrokoptéra komunikuje s uživatelem. Přes sériový port hlavního procesoru jsou připojeny další senzory na navigační desce a regulátory motorů. Deska senzorů obsahuje mikrokontrolér PIC (Microchip). Ten řídí ultrazvukový dálkoměr používaný k automatické výškové stabilizaci, a pomocí integrovaných A/D převodníků získává data z dvouosého gyroskopu Invensense IDG500. Dále čte data z přesnějšího gyroskopu Epson XV3700 na vertikální ose a tříosého akcelerometru Bosch BMA150 [5]. Komunikaci s regulátory motorů zajišťuje samostatná sériová linka, která se pomocí GPIO přepíná k jednotlivým regulátorům. Ty obsahují mikrokontrolér ATMega8a [15]. Jako akumulátor jsou použity tři sériově spojené Lithium-polymerové články o kapacitě 1000 mAh, pro bezpečnost uzavřené v plastovém pouzdře.

2.2 Původní účel AR.Drone

AR.Drone je primárně určena pro hraní her na platformách iPhone, iPad nebo iPod touch s takzvanou rozšířenou realitou (Augmented reality), kde se používá kombinace videa posílaného kvadrokoptérou s počítačem dokreslenými herními prvky. Tak jsou realizovány např. simulace vzájemných ozbrojených soubojů několika kvadrokoptér, závody apod. [14]. Zařízení je zamýšleno pro použití širokou veřejností a z toho pramení i vysoké nároky na bezpečnost. Jak je popsáno v [5] a [12], je AR.Drone je vybavena řadou bezpečnostních opatření, jako omezení maximální horizontální rychlosti letu, automatické zastavení při ztrátě Wi-Fi spojení, watchdog řídicí komunikace, který po 50 ms bez příchozího příkazu

zastaví kvadrokoptéru, a detekce kolize předmětu s listem vrtule, která způsobí okamžité vypnutí motorů a pád kvadrokoptéry. Detailní popis vnitřních senzorů a zabezpečení proti nehodám lze najít v článku [5].

AR.Drone se ve vzduchu automaticky stabilizuje pomocí interních senzorů. To zahrnuje automatické vyrovnávání náklonu a natočení kolem vertikální osy, udržování výšky a schopnost zastavení se na místě díky zpracování obrazu z vertikální kamery. Vzlet i přistání jsou vždy řízeny firmwarem. Ovládání je pak realizováno sadou vysokoúrovňových příkazů, ve kterých např. postačuje zadat pouze rychlosti v jednotlivých osách (vpřed/vlevo/nahoru).

Pro obohacení her umí AR.Drone provádět i tzv. letové animace a LED animace. Letové animace spočívají v provedení krátkého akrobatického manévru simulujícího postrčení, zásah nebo "havarování" kvadrokoptéry. Tyto relativně agresivní pohyby jsou uživatelem (resp. jeho aplikací) pouze aktivovány, dál jsou již plynule řízeny firmwarem, takže nedochází k přílišnému ohrožení okolí při výpadku spojení. LED animace využívají dvoubarevných kontrolních diod na každém z motorů, kterými vytvářejí světelné efekty.

Dalším prvkem podpory her je detekce barevných terčů nejen na ostatních kvadrokoptérách. Tyto terče jsou vidět např. na obrázku 2.2, tvoří je vždy výrazná oranžová a jedna ze tří dalších barev, ve kterých se AR.Drone dodává. Cílem je usnadnit herním aplikacím "zaměřování" protivníků a jiných cílů.

Obrázek 2.1: AR.Drone s krytem pro létání uvnitř budov. Naznačeny jsou směry otáčení vrtulí kvadrokoptéry.

Obrázek 2.2: AR.Drone s krytem pro venkovní let. Syté barvy umožňují detekci ostatními kvadrokoptérami.

2.3 Síťové služby

Aby bylo možné ovládat AR.Drone pomocí "chytrých telefonů", chová se zabudované Wi-Fi jako přístupový bod, ke kterému se telefony a jiné "ovladače" připojují stejně jako při přístupu k internetu. Po naběhnutí systému kvadrokoptéry se objeví veřejně dostupná síť s názvem `ardrone_NNNNNN` a funkčním DHCP serverem v podsíti `192.168.1.0/24`, adresa kvadrokoptéry je `192.168.1.1`. Uvnitř AR.Drone běží real-time operační systém na bázi Linuxu a je přístupný přes běžné telnetové spojení na portu 23. Sítě je využito i při aktualizaci firmwaru. Ta využívá FTP serveru běžícího na portu `5551`, kam uživatel nahraje soubor s aktualizací a odpojením napájení restartuje systém.

2.4 SDK

Aby mohly i třetí strany vyvíjet aplikace pro AR.Drone, je na stránkách [13] dostupné multiplatformní SDK. To mimo jiné obsahuje kostru řídicí aplikace pro Linux, kterou využívá zde popisovaný řídicí program. Z hlediska funkčnosti jsou v programu na straně SDK tři vlákna:

- *Řídicí* zajišťuje periodické volání funkce pro čtení dat ovladače (např. gamepadu).
- *Navigační* volá uživatelskou funkci vždy, když jsou přijata tzv. *navdata*, tj. struktura s daty o orientaci a stavu kvadrokoptéry.
- *Videostage*, které volá uživatelskou funkci po dekompresi snímku z kamery.

2.4.1 Ovládání

K ovládání kvadrokoptéry z řídicího vlákna stačí použít malou sadu funkcí SDK:

- `ardrone_at_set_progress_cmd()` je klíčovou funkcí pro řízení letu kvadrokoptéry. Čtyřmi parametry v rozsahu $\langle -1; 1 \rangle$ se nastavuje rychlost ve třech osách a rychlost rotace kolem svislé osy.
- `ardrone_tool_set_ui_pad_start()` způsobí automatický vzlet nebo přistání podle parametru.
- `ardrone_tool_set_ui_pad_select()` nouzově vypne motory a nechá kvadrokoptéru spadnout na zem. Aplikace by měla před odvoláním nouzového stavu požádat o přistání, jinak se obnoví původní stav letu a kvadrokoptéra může ihned nechtěně odstartovat.
- `ARDRONE_TOOL_CONFIGURATION_ADDEVENT()` je makro, kterým lze přepínat vnitřní nastavení firmwaru. Mezi parametry firmwaru je velmi vhodné nastavit maximální náklon kvadrokoptéry, který v podstatě odpovídá maximální dosažitelné rychlosti. Nabízen je také režim multikonfigurace, kde si může každá aplikace uložit vlastní sadu nastavení.
- `ardrone_at_set_flat_trim()` provede inicializaci inerciální jednotky při startu programu. V té chvíli by kvadrokoptéra měla ležet na rovném povrchu.
- `ardrone_at_navdata_demo()` přepne posílání *navdat* do běžného režimu, ve kterém se neposílají nadbytečné ladicí informace.
- `ardrone_at_set_navdata_all()` nastaví zmíněný režim na příjem všech dat.

Čtení stavu joysticku je rovněž v režii SDK a poskytuje vstup ve formě proporcionálních hodnot všech os v rozsahu $\langle -32768; 32767 \rangle$ a pravdivostních hodnot o stisku tlačítek. I když je práce s joysticky v Linuxu založena na zpracování událostí, jsou čteny jen v pravidelných intervalech v rámci funkce v řídicím vlákně. Vstup z joysticku lze téměř přímo použít pro volání zmíněných funkcí SDK a umožnit řídicí aplikaci manuální ovládání. Skutečná síťová komunikace s kvadrokoptérou, jejíž řídicí část je ve formě AT příkazů [12], je programátorovi skryta.

2.4.2 Navigační data

Přijímaná navigační data, v popisu SDK nazývaná *navdata*, poskytují informace o náklonu a otočení kvadrokoptéry, stavu řízení (letí/přistává/...), výšce nad zemí a úrovni nabití akumulátoru. Data jsou reprezentována strukturou v jazyce C a v ideálním případě přijímána každých 5 ms, což odpovídá frekvenci 200 Hz. Ne všechna měření jsou takto vzorkována, např. ultrazvukový dálkoměr pracuje na opakovací frekvenci 22,22 Hz příp. 25 Hz, podle nastavení. Pokud je nastavena detekce barevných terčíků v obraze, jsou v této struktuře obsažené jejich pozice

2.4.3 Video

Data z kamery jsou předávána jako sled RGB hodnot a jsou kompatibilní s reprezentací obrazu používanou knihovnou OpenCV (viz kapitola 5). Rozlišení obrazu je 320×240 px (QVGA) pro horizontální a 176×144 px (QCIF) pro vertikální kameru, vše ve snímkové frekvenci 15 Hz. Výběr kamery, jejíž obraz je odesílán uživateli se nastavuje jako parametr firmware výše zmíněným makrem. Dostupné jsou čtyři režimy – každá z kamer samostatně nebo kombinace obou, kde je vždy jeden z obrazů zmenšený v levém horním rohu.

Video je posíláno v komprimované podobě, podle [12] se používá jeden ze dvou možných video kodeků, UVLC (údajně podobný MJPEG) a P264 (podobný H.264). Obraz je také vnitřně zpracováván kvadrokoptérou, provádí se odhad dopředné rychlosti kvadrokoptéry, který se kombinuje s informacemi z vnitřního akcelerometru [5]. Výsledný odhad je přijímán v rámci struktury navigačních dat. Ač má obraz vertikální kamery relativně malé rozlišení, poskytuje firmwaru obraz ve snímkové frekvenci 60 Hz. Toho se kromě zpřesnění informací o rychlosti využívá ve stavu vznášení se na místě. Kvadrokoptéra by také měla mít schopnost vznášet se nad barevným terčem.

2.4.4 Kódování AT příkazů

I když je skutečná komunikace s AR.Drone přenechána SDK, je možné kvadrokoptéru řídit téměř jakýmkoli zařízením se schopností Wi-Fi připojení. Výše popsaná množina ovládacích funkcí téměř odpovídá sadě AT příkazů, které jsou skládány za sebe a posílány jako UDP pakety na port 5556. Každý příkaz začíná textem `AT*` a názvem příkazu, pokračuje znakem `'='`, za kterým následují datová pole oddělená čárkami a končí koncem řádku (UNIX-ového typu, tedy znakem #10). V prvním datovém poli se vždy přenáší sekvenční číslo příkazu, začínající od 1, v ostatních datových polích se přenášejí parametry příkazů. Číselné hodnoty přenášejí 32 bitová celá čísla se znaménkem zapsaná v dekadickém tvaru. Stejně jsou přenášena i bitová pole. Další možností jsou řetězce, které musí být označeny uvozovkami (znak #34). Desetinná čísla se pak posílají obdobně jako bitová pole, jejichž reprezentace ve formátu IEEE 754 je interpretována jako celé číslo se znaménkem, které je odesláno.

- `AT*PCMD` řídí směr pohybu. Po sekvenčním čísle následuje bitové pole a čtyři desetinné hodnoty popsané v rámci funkce `ardrone_at_set_progress_cmd()`. Bitové pole obsahuje bit 0, který povoluje pohyb kvadrokoptéry (ve stavu 0 je v režimu zastavení na místě), a bit 1 povolující pokročilejší styl pohybu, tzv. Combined yaw mode.

- `AT*REF` ovládá jak vzlet a přistání, tak nouzové vypnutí. V rámci jediného parametru (nepočítaje sekvenční číslo) se přenáší bitové pole, jehož bit 9 při hodnotě 1 indikuje požadavek, aby AR.Drone vzlétla případně při hodnotě 0 přistála. Nouzový stav se chováním mírně liší, bit 8 v hodnotě 1 způsobí pouze přepnutí nouzového stavu

a hodnota 0 je vysílána pouze k oddělení dvou změn stavu. Zbylé bity musí nabývat pevných hodnot, k hodnotě lze přičítat číslo 290717696.

- `AT*CONFIG` nastavuje parametry firmwaru. Po sekvenčním čísle následují dva řetězce, první identifikuje vlastnost, druhý jí přiřazuje hodnotu.
- `AT*FTRIM` inicializuje inerciální jednotku, obsahuje pouze sekvenční číslo.
- `AT*CONFIG_IDS` se používá v režimu multikonfigurace k identifikaci sady nastavení, se kterou pracuje příkaz `AT*CONFIG`. Parametry jsou sekvenční číslo a tři řetězce, které identifikují sezení, uživatele a aplikaci.
- `AT*COMWDG` resetuje příznak vypršení watchdogu, který je hlášený v navdatech a aktivuje se ve chvíli, kdy se mezi příchozími ovládacími pakety vyskytne interval delší než 50 ms.
- `AT*LED` aktivuje LED animaci. V parametrech po sekvenčním čísle následuje index animace, desetinné číslo představující frekvenci animace v Hz a délku trvání celé animace v celých sekundách.
- `AT*ANIM` aktivuje letovou animaci. Parametry jsou sekvenční číslo, index animace a celková doba v sekundách.

Podrobnější popis AT příkazů lze nalézt v [12].

2.5 Shrnutí

AR.Drone je kvadrokoptéra vybavená tříosým gyroskopem, akcelerometrem, ultrazvukovým výškoměrem a dvěma kamerami. Firmware kvadrokoptéry zajišťuje její automatickou stabilizaci ve vzduchu a je ovládán přes Wi-Fi síť pomocí malé sady popsaných AT příkazů z dostupného SDK. Příchozí data obsahují strukturu se stavem a orientací stroje, dostupný je také video stream z obou kamer.

Kapitola 3

Rozšíření senzorů o GPS

Pro účely autonomní navigace ve venkovním prostředí je zapotřebí znát aktuální pozici kvadrokoptéry. V dnešní době jsou dostupné samostatné GPS moduly, které stačí připojit ke zdroji napájení a přijímat již zpracovaná data o aktuální poloze. K navigaci kvadrokoptéry je ovšem velmi výhodné znát i její absolutní natočení vzhledem k Zemi, aby bylo možné nabrat správný směr k cíli. Samostatné GPS moduly ale neposkytují informaci o natočení, pokud se přijímač nepohybuje, případně ji poskytují velmi nepřesnou, pokud se pohybuje malou rychlostí. Důvodem je nestálá chyba v určení pozice mezi měřeními, která se běžně pohybuje kolem 10 m a způsobuje falešný pohyb i u stojícího přijímače. Směr pohybu tak nelze spolehlivě vypočítat. Jako lepší řešení se nabízí použití magnetického kompasu.

3.1 Modul GPS

Jako nejvhodnější modul byl zvolen *LS20126* od firmy LocoSys [10]. Je to totiž přijímač GPS, který kromě integrované antény obsahuje i magnetický kompas a akcelerometr v jediném modulu. Jak je u GPS přijímačů obvyklé, datová komunikace používá standard NMEA. Na hardwarové úrovni se s modulem komunikuje běžnou UART linkou, tato linka je obousměrná, je tedy možné posílat příkazy i směrem do modulu. Základní nastavení linky předpokládá 9600 baudů/s, 8 bitů, bez parity, s jedním stop bitem a bez řízení toku. Katalogový list je k dispozici na [9]. Kvůli možným problémům se stahováním a výskytem různých zastaralých verzí je taktéž přiložen v datové příloze práce.

3.1.1 Formát dat

Formát dat přijímaných z modulu vychází ze standardu NMEA 0183 [11]. Data přichází v textové podobě rozdělená do "vět" ukončených odřádkováním, každá věta nese specifickou podmnožinu dat podle svého typu. Příklad přijaté věty je vidět na obrázku 3.1. Každá věta je uvozena znakem `'$'`, a typem věty tvořeným pěti znaky – dvěma pro identifikaci odesílajícího zařízení (`GP` pro GPS) a dalšími třemi určujícími typ dat. Za nimi následuje pevný počet datových polí oddělených čárkami. Poslední datové pole věty je odděleno hvězdičkou, po které následují dva hexadecimální znaky kontrolního součtu a konec řádku. Kontrolní součet se počítá operací XOR přes všechny znaky věty mezi `'$'` a `'*'`.

3.1.2 Formát věty GPGGA

Informaci o poloze obsahuje věta typu $GPGGA, která poskytuje zeměpisnou šířku a délku, platnost zaměřené polohy, počet sledovaných satelitů a pravděpodobnou přesnost. Hodnoty jako čas a poloha, které přenášejí více hodnot v jednom poli (např. čas přenáší hodiny, minuty a vteřiny), jsou přenášeny s pevným počtem desetinných míst. Na obrázku 3.1 je čas ve formátu HHMMSS.SSS, zeměpisná šířka ve formátu SSMM.MMMM (stupně a minuty), a zeměpisná délka ve formátu SSSMM.MMMM. Je důležité nezaměnit velikost polí pro stupně – počet stupňů zeměpisné šířky nabývá hodnot do 90 oproti zeměpisné délce, která dosahuje až 180, ve větě tak mají vyhrazené dva resp. tři znaky. Platnost zaměřené pozice indikuje číselná hodnota. Popis platnosti pozice je detailněji popsán v [9], pro naše účely postačuje rozlišit stavy

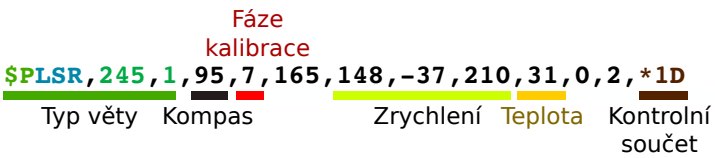
- 0 – žádná nebo neplatná pozice
- 1 – pozice je platná
- 6 – pouze odhad pozice po ztrátě signálu

Hodnota HDOP (Horizontal Dilution of Precision) udává přesnost aktuálního zaměření pozice jako násobek základní přesnosti GPS přijímače udávané výrobcem, např. hodnota HDOP=1,50 u přijímače s přesností ±5 m by měla odpovídat přesnosti ±7,5 m. Tato hodnota závisí na vzájemném uspořádání satelitů použitých pro výpočet pozice. O přesnosti určené pozice může napovídat i počet aktivních (použitých) satelitů při výpočtu.

Obrázek 3.1: Věta typu GPGGA přijatá z GPS modulu.

3.1.3 Formát věty PLSR

Protože standard NMEA dovoluje výrobcům definovat proprietární zprávy uvozené '$P' a třípísmenným kódem, je možné tímto způsobem posílat dodatečné informace specifické pro daný hardware. Modul LS20126 posílá věty s označením $PLSR (příklad na obrázku 3.2), které přenáší data z kompasu, podrobné hodnoty magnetického měření a zrychlení interního akcelerometru. Akcelerometr je bohužel pro účely navigace nepoužitelný, kvůli nízké frekvenci jeho aktualizace 1 Hz. Ve zmíněné větě je první blok dat vždy 245, druhý blok se liší (nabývá hodnot 1, 2, 7) podle typu přenášené informace. Klíčová položka pro tuto práci je pouze hodnota kompasu ve větě $PLSR,245,1 a fáze jeho kalibrace. Ta nabývá v průběhu kalibrace hodnot 0 až 7 – dokončeno.



Obrázek 3.2: Věta typu PLSR přijatá z GPS modulu.

3.1.4 Kalibrace kompasu

Magnetický senzor na modulu vyžaduje pro správné fungování kalibraci. Ta se spouští při každém zapnutí modulu a její fáze jsou indikovány ve výše popsané větě (význam fází není v katalogovém listu definován). Pro zkalibrování modulu je nutné pomalu otáčet zařízením postupně podle jednotlivých os, nebo jím mávat v rovinách XY, YZ a XZ. (Postup je popsán v [8], lze jej taktéž nalézt v datové příloze práce.) Podle katalogového listu [9] je pro uchování vnitřní paměti a hodin reálného času možné připojit zálohovací baterii. Jak je ale zmíněno dále v příloze B, je kalibrace přesto spouštěna po každém zapnutí modulu.

3.2 Propojení modulu s PC

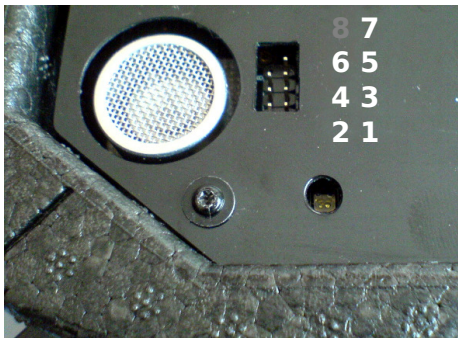
Řídicí aplikace na PC je schopna komunikovat s kvadrokoptérou AR.Drone pomocí Wi-Fi sítě (různé části komunikují na portech kolem 5555, viz [12]), je ale nutné získat data i z GPS modulu. Při návrhu bylo nejprve uvažováno o možnosti připevnění turistického GPS zařízení s připojením přes Bluetooth, kde je příznivá i hmotnost celého zařízení. Jako nedostačující se bohužel ukázal maximální dosah Bluetooth zabudovaného v takovýchto produktech, používá se totiž pouze třída 2 s dosahem do 10 m. Výrobky s delším dosahem nebo jiným typem bezdrátového spoje se buďto nevyrábí, nebo nejsou dostupné – v běžném komerčním/civilním použití ani nejsou potřeba. Nabízející se propojení zvoleného modulu *LS20126* s RF modulem o vyšším dosahu, např. Bluetooth třídy 1, je jednoduché, oddělené od hardware kvadrokoptéry a vyžaduje pouze napájecí zdroj, nicméně je to další rádiový spoj a potencionální zdroj rušení (bylo by nutné použít Bluetooth s vysokým vysílacím výkonem). Nevýhodou by byla i vyšší spotřeba, velikost a hmotnost. Pro napájení je možné zneužít konektor balanceru [1] , podle popisu rozšiřujícího konektoru kvadrokoptéry je k dispozici také 5 V na portu USB.

Po měření portu USB a hledání popisu na oficiálním fóru k AR.Drone projektu [13] se ukázalo, že USB port je zamýšlen pro budoucí hardwarová rozšíření, je však včetně napájení vypnut firmwarem. Zapojení rozšiřujícího konektoru je na stránkách projektu, s obvyklými vodiči pro USB a 12 V napětím z akumulátoru, zbylé dva piny jsou značené jako nepřipojené. Podle oficiálního fóra projektu jsou to však ve skutečnosti vodiče jednoho z UART portů hlavního procesoru (viz. obrázek 3.3), tento port je navíc v operačním systému AR.Drone přístupný jako standardní znakové zařízení `/dev/ttyPA0`. Sériové porty AR.Drone a GPS modulu jsou dokonce napěťově kompatibilní, GPS modul na úrovních 3 V logiky, hlavní procesor na 3,3 V. Modul GPS tak lze připojit přímo na tyto vodiče. Řídicí aplikace potom může vytvořit TCP spojení na portu telnetu a pomocí odeslání krátkého skriptu s nastavením parametrů přenosu začít přijímat data. Takto zcela odpadá nutnost dalšího vysílače.

3.2.1 Zdroj napájení

Přestože *LS20126* zvládá napájecí napětí až do 6 V, kvůli zmíněnému odpojení USB portu bylo nutné brát napájení z baterie a navrhnout tištěný spoj se stabilizátorem napětí (obrázek 3.4). Zapojení odpovídá referenčnímu schématu z katalogového listu modulu, vyvedeno

[1] Baterie s chemií Li-Pol a podobné jsou velmi citlivé na teplo, které se v nich hromadí při přebíjení. V sadách více článků se kromě silových "vybíjecích" vodičů zapojují ještě slabší "nabíjecí" konektory s vývody z míst mezi jednotlivými články pro balancer, který vyrovnává napěťové rozdíly mezi články aby se slabší článek nenabil dříve než ostatní a nepřebíjel se.



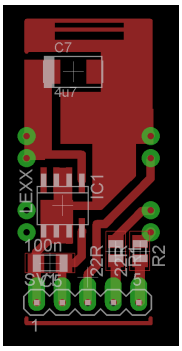
pin	označení	funkce
1	$5\,V$	napájení USB
2	$12\,V$	kladný pól baterie
3	D−	USB data
4	RX	příjem UART dat
5	D+	USB data
6	TX	vysílání UART dat
7	GND	záporný pól baterie
8	-	chybějící pin

Obrázek 3.3: Číslování a zapojení pinů rozšiřujícího konektoru.

je i napájení záložní baterie. Jako stabilizátor je použit LE33 (3,3 V). Připojený modul umístěný nad horizontální kamerou AR.Drone, kde má anténa výhled na oblohu, je na obrázku 3.5.

3.2.2 Reakce firmware na připojení GPS

Komplikace přímého připojení přes UART port způsobuje část firmware, která při příjmu dat na této sériové lince deaktivuje vnitřní řídicí program (pravděpodobně v domnění, že jde o aktualizaci firmware). Ten je nutné znovu spustit příkazem `/bin/program.elf` (případně před připojením modulu ukončit skript `/bin/check_update.sh`). I když by to mohl zařídit zmíněný skript pro nastavení parametrů přenosu dat, je nakládání se systémovými procesy v aktuální verzi přenecháno výhradně uživateli.



Obrázek 3.4: Plošný spoj se stabilizátorem pro napájení GPS modulu.

Obrázek 3.5: Modul LS20126 umístěný na kvadrokoptéře.

3.3 Shrnutí

Vybraný modul LS20126 byl zvolen díky přítomnosti magnetického kompasu a jednoduchému zpřístupnění jeho hodnot. Výstupní data jsou přenášena běžnou UART linkou ve formátu podle standardu NMEA. Nestandardním využitím volných prostředků na AR.Drone bylo možné dosáhnout přenosu těchto dat po Wi-Fi síti spolu s ostatní komunikací, tedy bez nutnosti použití samostatného vysílače. Pro napájení modulu bylo nutné navrhnout desku plošných spojů se stabilizátorem napětí.

Kapitola 4

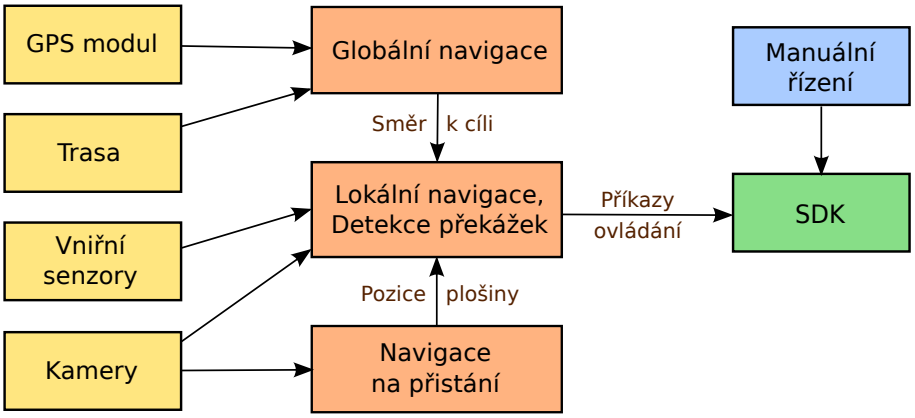
Navigační architektura

Navigační architektura popisuje rozložení řešeného navigačního problému na funkční bloky, které řeší dílčí úkoly problému. Jejich vzájemné propojení, reprezentované orientovanými vztahy, značí směr toku dat a jejich typ. Samotné bloky pak transformují data a předávají je dále. Výhodou tohoto přístupu je kromě zjednodušení návrhu i lepší lokalizace případného problému a možnost odděleného testování. Pro účely této práce je třeba rozlišit několik nezávislých úkolů, jako je sledování trasy a hledání přistávací plošiny podle kamer. Podstatné je také předzpracování dat poskytovaných ze senzorů.

Navržená navigační architektura se skládá ze zdrojů dat (senzorů) a z bloků, které je zpracovávají. Senzory, které jsou na obrázku 4.1 zobrazeny žlutě, zastupují zdroje dat poskytované SDK (video, navdata), data z GPS modulu a souřadnice trasy z textového souboru. Oranžově jsou označena místa zpracování dat. Posledními bloky je část kódu v SDK a vstup ovládání z joysticku. Funkce bloků zpracovávajících data jsou následující.

- *Globální navigace* přijímá aktuální pozici a podle souboru trasy vypočítává azimut a vzdálenost k dalšímu podcíli cesty. Při tom kontroluje kvalitu signálu a dobu od posledního přijetí platných dat, a případně signalizuje chybu.
- *Navigace na přistání* slouží pouze k vyhledávání přistávací plošiny a předává informace o její pozici v obraze. Poskytuje zvlášť informace z horizontální a vertikální kamery.
- *Lokální navigace* pak přebírá výsledky *Globální navigace* a *Navigace na přistání*, ze kterých vyhodnocuje směr cesty a vypočítává řízení podle aktuální fáze letu. Mimo to je napojena na vstup obrazu z kamer, snaží se rozpoznat možné překážky a vyhnout se jim.
- *SDK* je označena úprava části kódu v řídicím vlákně, která přijímá manuální řízení a podle požadavku uživatele povoluje autonomní řízení. Nutnost povolení autonomního řízení uživatelem je navržena primárně jako bezpečnostní prvek v případě jeho selhání.

Situace vyžadující prioritní řízení jako vyhýbání se překážkám jsou řešeny *"lokálními manévry"*, které buď upravují běžné chování, nebo ve významných situacích zcela přebírají řízení. Příkladem je detekování překážky s malým ohodnocením, které pouze upraví aktuální hodnoty ovládání tak, aby se kvadrokoptéra mírně odchýlila z kurzu. Nalezení přistávací plošiny spodní kamerou na konci trasy naopak způsobí převzetí celého řízení přistávacím manévrem.



Obrázek 4.1: Navigační architektura řídicího programu. Senzory jsou zobrazeny žlutě, výpočetní bloky oranžově. Část kódu odpovědná za řízení v SDK je označena zeleně, její vstup z joysticku modře.

4.1 Filtrace globální pozice

Protože aplikace využívá GPS na hranici její přesnosti, je vhodné pozici filtrovat proti šumu. K tomu je použit Kalmanův filtr, který je přiblížen v [4]. V této aplikaci je třeba definovat stavový vektor x_k, matici přechodu F, která zohledňuje pohyb mezi aktualizacemi a další pomocné matice, samotný výpočet je pak zajišťován knihovnou OpenCV (viz podkapitola 6.4). Kalmanův filtr umožňuje jednak potlačit chyby měření pozice a také do jisté míry předpovědět pozici kvadrokoptéry před dalším měřením.

Stavový vektor x_k obsahuje dvě souřadnice polohy kvadrokoptéry x, y a rychlost pohybu v_x, v_y. Pozice x značí vzdálenost bodu aktuální pozice od nultého poledníku v kilometrech, měřenou podél rovnoběžky probíhající tímto bodem, obdobně y značí vzdálenost tohoto bodu od rovníku. Matice přechodu F předpovídá stav od poslední aktualizace podle rovnice 4.1, kde $w_k \sim N(0, R_k)$ je šum procesu (předpovědi). Měření skutečného stavu ve chvíli aktualizace poskytuje pouze podmnožinu údajů stavového vektoru v podobě vektoru z_k (pouze polohu, ne rychlost). Měření stavu odpovídá rovnici 4.2, kde $v_k \sim N(0, Q_k)$ je šum měření. Hodnoty kovariančních matic byly vyzkoušeny ručně a jsou nastaveny podle rovnice 4.3. Q_k značí kovarianční matici šumu procesu a R_k kovarianční matici šumu měření, přičemž h je relativní přesnost hlášená GPS modulem.

$$x_k = Fx_{k-1} + w_k, \qquad \text{kde } x_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_k, \qquad F = \begin{bmatrix} 1 & 0 & dx & 0 \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

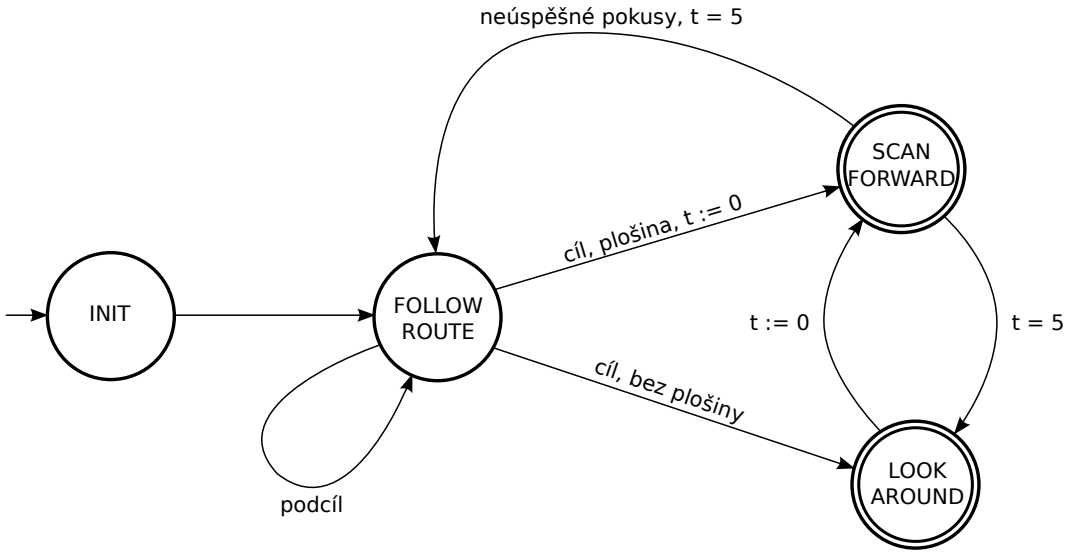
$$z_k = H^T x_k + v_k, \qquad \text{kde } z_k = \begin{bmatrix} z_x \\ z_y \end{bmatrix}_k, \qquad H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.2}$$

$$Q_k = \begin{bmatrix} 0{,}1 & 0 & 0 & 0 \\ 0 & 0{,}1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}, \qquad R_k = \begin{bmatrix} 10^3h & 0 & 0 & 0 \\ 0 & 10^3h & 0 & 0 \\ 0 & 0 & 10^3h & 0 \\ 0 & 0 & 0 & 10^3h \end{bmatrix} \tag{4.3}$$

4.2 Fáze letu

Fáze letu, které řídí chování bloku *Lokální navigace*, jsou modelovány časovaným automatem [1] podle obrázku 4.2. Použití časovaného automatu umožňuje popisovat řízení v reálném čase a omezit tak délku trvání některých fází letu. Jinak by program mohl setrvat ve stavech neomezeně dlouho a způsobit nechtěné chování kvadrokoptéry.

V počátečním stavu `INIT` probíhá nezbytné čekání na inicializaci (kalibraci) magnetického kompasu a na určení přesné globální pozice GPS, bez níž je zbytečné vzlétnout. Po splnění požadavků následuje stav sledování trasy `FOLLOW_ROUTE`, ve kterém se letí v preferovaném směru (azimutu) vypočteném *Globální navigací*, s lokálními korekcemi v případě výskytu překážek. Při dosažení cílového bodu se chování rozlišuje podle aktuální viditelnosti přistávací plošiny. Bez viditelné plošiny automat přechází do stavu rozhlédnutí se `LOOK_AROUND`. Ten slouží k pozorování a ohodnocení terénu v okolí aktuální pozice. Kvadrokoptéra rotuje na místě, přičemž blok *Navigace na plošinu* hledá směr s maximálním ohodnocením. Poté přejde do stavu `SCAN_FORWARD`, kdy se kvadrokoptéra vydá nejvhodnějším směrem kupředu. Do tohoto stavu se může dostat i přímo ze stavu `FOLLOW_ROUTE`, pokud je ve chvíli přiblížení se k cíli viditelná plošina v obraze.



Obrázek 4.2: Časovaný automat fází letu.

Cílem fáze `SCAN_FORWARD` je přiblížit se k domnělé plošině a zabrat ji pomocí vertikální kamery, která určí, zda se opravdu jedná o přistávací plošinu nebo ne. Zde se uplatňuje zmíněná vlastnost časovaného automatu – trvání této fáze je omezeno na maximálně 5 s. Po vypršení tohoto času automat přechází do stavu ohodnocení okolí `LOOK_AROUND`. Aby se předešlo "odlákání" kvadrokoptéry od cílové GPS souřadnice např. vzdálenou červenou plochou, je stanoven maximální limit opakování po sobě jdoucích fází rozhlédnutí a skenování. Po překročení počtu iterací se restartuje blok *Globální navigace* do stavu sledování posledního (cílového) bodu trasy.

Počet zmíněných iterací je malý (pouze dvě, viz podkapitola 7.3). Kvadrokoptéra se opakovaně vrací na GPS souřadnici cíle, odkud (pokud nenajde přistávací plošinu) pokračuje pod úhlem o 45° větším. Pokud tedy není přistávací plošina vůbec detekována, připomíná trasa hledání hvězdici se středem v cílovém bodě. Podrobnější popis řízení ve všech stavech je popsán v následující podkapitole.

4.3 Návrh řízení ve stavech časovaného automatu

V této části je popsáno řízení v jednotlivých stavech fází letu, jak je vypočítává blok *Lokální navigace*. Všechny hodnoty řízení leží v intervalu $\langle -1; 1 \rangle$, záporná hodnota v ose X znamená pohyb vlevo, záporná hodnota v ose Y značí pohyb vpřed (podle formátu AT příkazů v [12]).

`INIT` přechází do dalšího stavu ve chvíli, kdy přesnost globální pozice přestane být zavádějící a kompas hlásí, že je zkalibrován. Do té doby je řízení v klidu (všechny hodnoty řízení jsou nulové) a kvadrokoptéra šetří baterii setrváním na zemi.

`FOLLOW_ROUTE` sleduje trasu tím, že zabezpečuje otočení kvadrokoptéry do směru určeného globální navigací a způsobuje její pohyb vpřed směrem k podcíli trasy. Řízení je při tom navrženo proporcionálně, aby se minimalizovaly ostré změny. Rychlost otáčení je při rozdílu aktuálního a požadovaného úhlu nad 30° maximální, pod touto hranicí (v blízkosti požadovaného směru) pak klesá lineárně k nule. Řízení je dáno následujícími rovnicemi, kde α je rozdíl mezi preferovaným směrem cesty a aktuálním otočením kvadrokoptéry ve stupních, $v_X(\alpha)$ a $v_Y(\alpha)$ jsou hodnoty řízení pro horizontální rychlost a $v_R(\alpha)$ je rychlost rotace kolem svislé osy.

$$v_X(\alpha) = \begin{cases} 0 & \text{pro } |\alpha| \geq 30°, \\ sin(\alpha \cdot \pi/180) & \text{pro } |\alpha| < 30° \end{cases} \tag{4.4}$$

$$v_Y(\alpha) = \begin{cases} 0 & \text{pro } |\alpha| \geq 30°, \\ -cos(\alpha \cdot \pi/180) & \text{pro } |\alpha| < 30° \end{cases} \tag{4.5}$$

$$v_R(\alpha) = \begin{cases} 1 & \text{pro } \alpha \geq 30°, \\ -1 & \text{pro } \alpha \leq -30°, \\ \alpha/30 & \text{pro } |\alpha| < 30° \end{cases} \tag{4.6}$$

Hlášení blízkosti cílového bodu trasy *Globální navigací* a údaje o zbývající vzdálenosti se využívá k aktivaci bloku *Navigace na přistání*. Podle toho je také přepnuta hodnota v nastavení firmwaru AR.Drone, která zajistí přenos obrazu z vertikální kamery. Aktivace ještě před dosažením cílového bodu (podle GPS) řeší případ, kdy kvadrokoptéra ještě před dosažením cílové souřadnice proletí kolem plošiny a musela by se později vracet. Přechod z fáze `FOLLOW_ROUTE` je závislý na dosažení dostatečně malé vzdálenosti od cílového bodu.

`LOOK_AROUND` fázi řídí jednoduchý stavový automat, který zajišťuje plynulé otočení o 360°, přičemž se uchovává úhel, pod kterým bylo dosaženo nejvyššího ohodnocení obrazu horizontální kamery. Na tento úhel se poté kvadrokoptéra natočí a fáze končí. Plynulý začátek a konec otáčky je zajištěn rychlostí otáčení závislou na rozdílu aktuálního úhlu natočení a počátečního/cílového úhlu obratu. Velikost rozdílu ve stupních je značena α.

$$v_R(\alpha) = min\left(\frac{1}{4} + \left|\frac{\alpha}{30}\right|, \quad 1\right) \tag{4.7}$$

`SCAN_FORWARD` využívá k výpočtu řízení pouze *horizontální* kameru, podle které se přibližuje k přistávací plošině. Tento stav se chová odlišně podle toho, zda je plošina aktuálně viditelná.

- Pokud ano, snaží se dostat detekovaný tvar do středu obrazu pomocí otočení kvadrokoptéry. Pozici tvaru v obraze z *horizontální* kamery hlásí *Navigace na přistání* jako číslo v intervalu $\langle -1; 1 \rangle$ (vlevo až vpravo). To je důležité rozlišovat od detekce na *vertikální* kameře, která předává pozici jako vektor a poskytuje tak dvojici souřadnic x a y. Pokud je plošina přímo proti kvadrokoptéře, tedy hodnota x je blízko nuly, řízení lineárně zvyšuje dopřednou rychlost. Aby se předešlo nesprávnému chování v situaci, kdy je kvadrokoptéra natolik blízko plošině, že se ztrácí z obrazu, je pevně zadána dopředná rychlost.
- V případě, kdy plošina není v obraze viditelná, je řízení nastaveno vpřed pevně zadanou rychlostí pro průzkum okolí.

Fáze končí neúspěšně vypršením času, nebo úspěšně v prvním okamžiku, kdy je *vertikální* kamerou detekována přistávací plošina (začíná přistávací manévr `LAND`). Řízení v tomto stavu využívá pouze dopřednou rychlost $v_Y(x)$ a rotaci $v_R(x)$, chová se podle podle rovnic 4.8 a 4.9. Pozice plošiny na *horizontální* kameře je značena x, přítomnost a blízkost plošiny značí pořadě logické proměnné P a C.

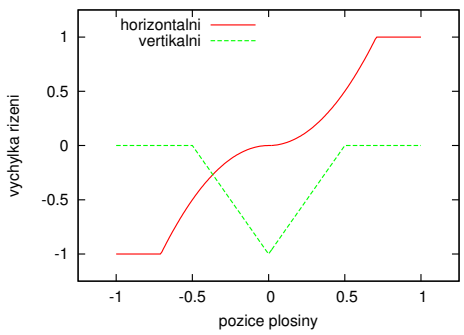
$$v_Y(x) = \begin{cases} min\,(2 \cdot |x| - 1, \quad 0) & \text{pro } P \wedge \neg C, \\ -0{,}25 & \text{pro } P \wedge C, \\ -0{,}5 & \text{pro } \neg P \end{cases} \tag{4.8}$$

$$v_R(x) = \begin{cases} x & \text{pro } P \vee C, \\ 0 & \text{pro } \neg P \wedge \neg C \end{cases} \tag{4.9}$$

4.4 Lokální manévry

Součástí řídicího programu jsou manévry pro dosednutí na přistávací plošinu, zotavení se z neúspěšného pokusu o přistání a úhybný manévr. Tyto manévry jsou reakcí na situace, které vyžadují neprodlené řešení.

`LAND` je manévr aktivovaný kdykoli *vertikální* kamera zachytí plošinu. K aktivaci manévru během sledování trasy nedojde nechtěně, jelikož je detekce plošiny aktivována až v blízkosti cíle. Řízení je obdobné jako ve fázi `SCAN_FORWARD`, kvůli vyšší prioritě je ale navrženo jako manévr, který převezme řízení. Na rozdíl od ní také využívá informace pouze z *vertikální* kamery, pozice plošiny je tedy předávána jako vektor se složkami x a y. Manévr se pokouší dostat detekovaný objekt do středu obrazu, při jehož dosažení začne klesat. Řízení v závislosti na pozici plošiny (její vzdálenosti od středu) je popsáno následujícími funkcemi a je taktéž ilustrováno na obrázku 4.3.



Obrázek 4.3: Průběh výchylek řízení v závislosti na pozici plošiny.

$$v_X(x) = \begin{cases} -1 & \text{pro } x \in \left\langle -1; -\frac{\sqrt{2}}{2} \right\rangle, \\ sgn(x) \cdot 2x^2 & \text{pro } x \in \left(-\frac{\sqrt{2}}{2}; \frac{\sqrt{2}}{2} \right), \\ 1 & \text{pro } x \in \left\langle \frac{\sqrt{2}}{2}; 1 \right\rangle \end{cases} \tag{4.10}$$

$$v_Z(r) = \begin{cases} 0 & \text{pro } r \in \langle -1; -0,5 \rangle, \\ 2 \cdot |r| - 1 & \text{pro } r \in (-0,5; 0,5), \\ 0 & \text{pro } r \in \langle 0,5; 1 \rangle \end{cases} \tag{4.11}$$

Rovnice 4.10 popisuje řízení v ose X v závislosti na pozici plošiny; stejná rovnice platí i pro osu Y. Rovnice 4.11 popisuje řízení ve svislé ose v závislosti na vzdálenosti plošiny r, kde $r = \sqrt{x^2 + y^2}$. Řídicí program s takto nastaveným řízením je schopen přistát na střed plošiny o rozměru 1,25×1,25 m ve venkovním prostředí i za mírného větru. Z praktické zkušenosti z manuálního řízení je od výšky přibližně 40 cm nad povrchem plošiny obtížné udržet se nad jejím středem. V této výšce se tedy vyplatí spuštění automatizovaného přistání, řešeného firmwarem, které je rychlejší a dosedne přesněji.

`GAIN_ALTITUDE` je čistě pomocný manévr aktivovaný po neúspěšném pokusu o přistání na plošinu, když je manévr `LAND` ještě před dosednutím na zem ukončen z důvodu zmizení plošiny z vertikální kamery. Jeho účelem je nastoupání zpět do výšky vhodné jak pro let, tak pro zpracování obrazu z kamer. Řízení je při tom fixně nastaveno na maximální rychlost stoupání.

`AVOID_COLLISION` se vyhýbá rozpoznaným překážkám tím, že k dopřednému pohybu přičítá pohyb stranou podle pozice a ohodnocení překážky. V případě silněji ohodnocené překážky pohyb vpřed zpomalí, nebo zcela zastaví. Výpočet řízení nejprve určuje směr, kterým je vhodné překážku obletět. Pozice detekovaných překážek jsou průměrovány s váhou podle jejich ohodnocení. Dále je porovnáno jejich celkové ohodnocení s referenční hodnotou. Rychlost pohybu stranou nabývá 50% povolené hodnoty při téměř nulovém ohodnocení a dosahuje 100% již na polovině referenční hodnoty. Z dopředné rychlosti je ponecháváno 100% z původní hodnoty při nulovém ohodnocení, dále je pak rychlost snižována až na 0%. Po začátku úhybného manévru je také nastavena minimální doba, po kterou je manévr udržován aktivní, pokud mezitím nebyla zjištěna jiná větší překážka. V případě nutnosti zastavení je tak řízení stále nastaveno na pohyb stranou. Předpokládá se, že časový interval bude dostačující k obletění překážky. Rovnice 4.12 popisuje novou rychlost $v'_X(v_X, s, d)$ na základě původní rychlosti v_X, ohodnocení překážky $s \in \langle 0; 1 \rangle$ a její pozici (vlevo/vpravo) $d \in \{-1; 1\}$. Druhá rovnice 4.13 vyjadřuje upravenou rychlost $v'_Y(v_Y, s)$ vypočtenou z ohodnocení překážky a původní rychlosti v_Y.

$$v'_X(v_X, s, d) = \begin{cases} v_X + d \cdot (0,5 + s) & \text{pro } s \in \langle 0; 0,5 \rangle, \\ v_X + d & \text{pro } s \in (0,5; 1\rangle, \end{cases} \tag{4.12}$$

$$v'_Y(v_Y, s) = v_Y \cdot (1 - s) \tag{4.13}$$

4.5 Shrnutí

Navigační architektura obsahuje čtyři senzory a tři bloky zpracovávající data. *Globální navigace* hlásí preferovaný směr k cíli, *Navigace na přistání* hlásí pozici plošiny. Tato data využívá *Lokální navigace*, která vypočítává řízení a při tom se chová podle fází letu popsaných časovaným automatem. Po fázi sledování cesty střídavě následují fáze rozhlédnutí se a sledování červeného objektu. Důležité situace jako finální přistávání a vyhýbání se překážkám řeší lokální manévry.

Kapitola 5

Zpracování obrazu

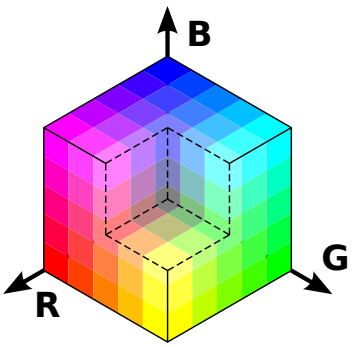
Řídicí program musí obzvláště při přistání zpracovávat obraz z kamer. Navigace kvadrokoptér pouze podle kamer je již zkoumána v článku [6]. Ten se zabývá autonomní navigací malých vrtulníků (včetně zde použité kvadrokoptéry AR.Drone) uvnitř budov. Zpracování obrazu při tom rozlišuje různé typy prostředí jako chodby, kanceláře a schodiště, ve kterých se zaměřuje na jejich specifické vlastnosti. Typickou vlastností chodeb je sbíhání většiny čar v obraze do úběžníku, jehož pozice odpovídá konci chodby. Dalším případem jsou schody, které jsou rozpoznány jako množství horizontálních čar v obraze. V článku popisované řešení je schopné navigace v chodbách, zatáčení v jejich rozích i pohybu vzhůru po schodišti.

Detekcí překážek v podmínkách podobných jako v této práci se zabývají články [3] a [7]. Oba přístupy využívají obrazu z jedné pohybující se kamery. Článek [7] popisuje řízení kvadrokoptéry ve virtuálním prostředí, v rámci něhož popisuje detekci překážek založenou na velikosti vektorů tzv. optického toku (viz podkapitola 5.3). Druhý článek [3] využívá segmentace trojrozměrných rovin vypočtených z optického toku (metoda vychází z [2]). Výsledkem jsou roviny obvykle se vyskytující v městském prostředí jako silnice a stěny okolních budov. Netypické roviny jsou pak považovány za překážky.

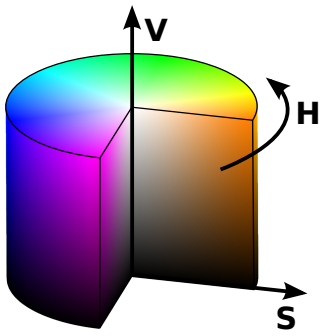
5.1 Barevné prostory

Klasická reprezentace RGB obrazu v rastrovém formátu využívá tří čísel, která představují intenzitu každé ze tří základních barev. Aditivním smícháním se dosáhne požadované barvy, kterou vnímá lidské oko. Na obrázku 5.1 je znázorněna krychle představující barevný prostor RGB, jednotlivé osy krychle odpovídají barvám R, G a B. Na souřadnici v počátku souřadnic (0, 0, 0), kde je intenzita všech složek nulová, je černá barva, naopak v maximu v bodě (1, 1, 1) je bílá atd.

Člověku je však přirozenější pracovat s barvou především podle odstínu, sytosti barvy, případně světelnosti. K tomu se využívá barevného prostoru HSV. Ten je na rozdíl od krychle mapován na válec (obrázek 5.2), jehož výška odpovídá světelnosti (value V) s černou dole a bílou nahoře, sytost pak vzdálenosti od osy rotace (nejsytější barvy jsou na stěně válce). Po obvodu válce se nakonec volí barva, kde za fialovou plynule následuje červená. Tento styl reprezentace se hodí nejen pro grafiky, ale je vhodný i na práci s barevnými odstíny v obrazu. Rozpoznání odstínu barvy nezávisle na její sytosti a světlosti je využito při detekci přistávací plošiny v následující podkapitole.



Obrázek 5.1: Barevný prostor RGB. Převzato z [19].



Obrázek 5.2: Barevný prostor HSV. Převzato z [19].

5.2 Detekce přistávací plošiny

Obraz z kamer přichází v rozlišení 320×240 px, v režimu hledání přistávací plošiny navíc obsahuje v levém horním rohu výřez o velikosti 88×72 px s obrazem z vertikální kamery, jak je možné vidět na obrázcích 7.8 a 7.5.

Protože testovací prostředí je přírodní a přistávací plošina je označena červenou barvou, která se obzvláště na zemi prakticky nevyskytuje, zaměřuje se detekční metoda na vyhledávání dostatečně velkého červeného objektu. Celý přijatý snímek je konvertován do barevného prostoru HSV a prahován podle odstínu, čímž jsou potlačeny předměty ostatních barev. Obraz je následně prahován ještě v závislosti na sytosti barvy a světlosti, pro odstranění méně sytých barev a stínů. Výsledkem je binární maska.

Další fází zpracování je počítání nenulových pixelů v obraze. Celkový počet přesahující zvolenou hranici indikuje přítomnost plošiny. Dále se využívá podíl světlé plochy v obraze pro ohodnocení snímku při fázi `LOOK_AROUND`. Při tom jsou oblasti obrazu blíže středu váhovány tak, aby se dosáhlo největšího ohodnocení v úhlu, ve kterém kvadrokoptéra směřuje přímo k plošině. Bez této úpravy by bylo skóre téměř konstantní po celou dobu, kdy je plošina v záběru. Výpočet je popsán rovnicí 5.1, kde w a h jsou rozměry obrazu, $p(x, y)$ funkce barvy pixelu a $m(x)$ je váhovací funkce podle rovnice 5.2. Konstanta r určuje vzdálenost od středu obrazu, ve které jsou váhy pixelů vyšší.

$$score = \sum_{x=0}^{w} \sum_{y=0}^{h} p(x, y) \cdot m(x) \tag{5.1}$$

$$m(x) = \begin{cases} 6 & \text{pro } \left|x - \frac{w}{2}\right| < r, \\ 1 & \text{jinde} \end{cases} \tag{5.2}$$

Nakonec je z rozpoznaného tvaru vypočteno těžiště.To je spolu s ohodnocením předáváno bloku *Lokální navigace* zvlášť pro vertikální kameru (informace o pozici je vektor) a zvlášť pro horizontální kameru, kde je předávána pouze jeho složka x. V obrazu horizontální kamery se dodatečně detekuje blízkost plošiny. Při tom se zjišťuje počet nenulových pixelů v několika spodních řádcích obrazu, který je porovnáván s referenční hodnotou. Pokud je hodnota nad limitem, je pravděpodobné, že část plošiny je již mimo záběr kamery.

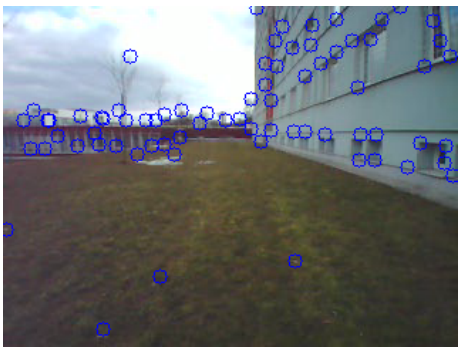
5.3 Detekce překážek

Princip detekce překážek použitý v této práci je založený na velikosti "změn" v obraze. Blízké překážky při pohybu kamery totiž způsobují větší změny než vzdálené pozadí, je tak možné detekovat kolemjdoucí chodce a jiné překážky. Řídicí program se jim následně může vyhýbat. To ovšem znamená, že kamera na místě nezjistí přítomnost překážky, dokud se nezačne pohybovat. Je tedy třeba se spolehnout na to, že se kvadrokoptéra překážce vyhne dříve, než bude nucena zastavit, případně je nutné řešit pohyby na místě, které poslouží k ohodnocení okolí, například vylétnutí nahoru a dolů.

Pro detekci větších pohybů se v této práci používá výpočtu optického toku [4]. Tato technika je schopna vypočítat pohyby vhodných bodů mezi dvěma podobnými snímky, v tomto případě mezi dvěma následujícími snímky videa z AR.Drone. Ne všechny body v obraze jsou ale vhodné ke sledování, například bod na bílé stěně při pohybu ztratí i lidské oko. Ve dvojrozměrných obrazech nejsou vhodné ani rovné hrany, protože na nich není pozorovatelný s nimi rovnoběžný pohyb, viz tzv. *Aperture problem*, vysvětluje jej např. [4].

V obraze se obvykle nejprve vyhledávají tzv. *rohy* (corners), které mají vhodný tvar ke sledování, lze je matematicky popsat a v obraze algoritmicky nalézt. Takovým algoritmem disponuje knihovna OpenCV, je navíc schopna dopočítat, i umístění těchto rohů s přesností menší než obrazový bod. Vhodné rohy ke sledování jsou ilustrovány na obrázku 5.3.

Nepříjemností v práci s obrazem jsou pohyby kamery vznikající manévrováním kvadrokoptéry. Samotné otáčení na místě kolem svislé osy například způsobuje téměř úplné selhání detekce optického toku. Další pohyby způsobují znatelné změny u všech vektorů, a úspěšně maskují vlastnosti toku typické pro překážky. Mnohé z nich jako celkový posuv a rotace obrazu lze matematicky popsat. Důležitou schopností je pak odečtení, nebo alespoň dostatečné potlačení takových pohybů.



Obrázek 5.3: Rohy vhodné ke sledování metodou optického toku.

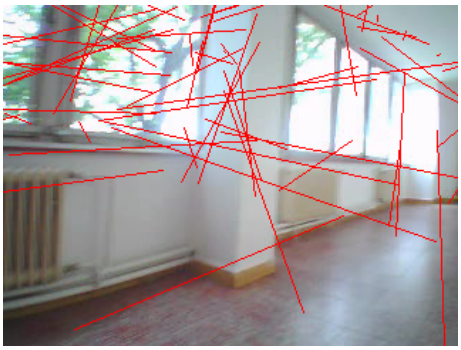
Obrázek 5.4: Příklad vektorů optického toku.

Pokud jsou rušivé jevy z vypočtených vektorů toku odstraněny, můžeme předpokládat, že vektory s větší délkou odpovídají předmětům v blízkosti kvadrokoptéry. Tyto výskyty je pak nutné vhodně sumarizovat, aby bylo možné spolehlivě rozhodnout o přítomnosti překážek a naplánovat úhybné manévry. Výsledek výpočtu optického toku je vidět na obrázku 5.4, vektory jsou vyznačeny červeně. Pro snazší viditelnost je tady, i v dalších ilustracích jejich délka třikrát prodloužena.

5.4 Potlačení pohybů kvadrokoptéry

Nejrušivějším pohybem je rotace kolem svislé osy. Pravděpodobně kvůli relativně velké úhlové rychlosti při otáčení a snímkové frekvenci 15 Hz metoda optického toku selhává – je přítomno příliš mnoho chybně vypočtených vektorů (jak ilustruje obrázek 5.5). Manévr způsobující rychlé otočení kolem této osy je nicméně vždy prováděn na místě. Protože při tomto stylu pohybu (bez horizontální rychlosti) detekční metoda z principu nefunguje a kvadrokoptéra jen s malou pravděpodobností do něčeho narazí, může být detekce překážek deaktivována. Tak je navržen i řídicí program, který od nastavené rychlosti otáčení zastavuje detekci překážek.

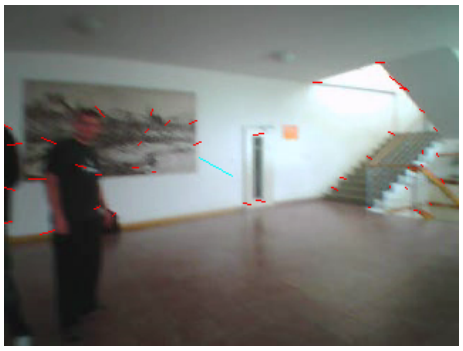
Znatelně menší chybu způsobují rotace kolem ostatních os kamery, kdy se kvadrokoptéra jen mírně naklání, aby získala horizontální zrychlení. Nezpůsobují selhání výpočtu optického toku, nicméně zapříčiní jednotný posun v obraze (obrázek 5.6).



Obrázek 5.5: Ukázka selhání výpočtu optického toku způsobená rychlým otočením kamery.

Původní přístup počítal s využitím navigačních dat pro zjištění rozdílů v orientaci kvadrokoptéry mezi snímky, bohužel tato data se při testování ukázala jako ne zcela synchronní, nebylo je tedy možné spolehlivě využít. Pravděpodobnou příčinou je nespolehlivost bezdrátového přenosu. Odečítání pohybů kvadrokoptéry se tak spoléhá pouze na obrazová data. Pro potlačení celkového pohybu v obraze je z platných vektorů optického toku vypočítán průměrný vektor, který je od nich odečten. Vyčištění obrazu je viditelné na obrázcích 5.6 a 5.7, kde modrá úsečka uprostřed zobrazuje průměr vektorů před odečtem. Na druhém obrázku je možné vidět, že délky vektorů v popředí jsou oproti pozadí delší. Délky všech vektorů jsou pro lepší viditelnost opět třikrát prodlouženy.

Obrázek 5.6: Pohyb kamery způsobující celkový posun obrazu.



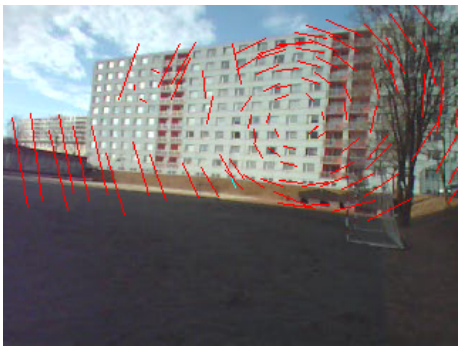
Obrázek 5.7: Totéž s odečteným průměrem (zobrazen modře).

Dalším nežádoucím pohybem kamery je rotace kolem osy jejího pohledu, která se promítá do obrazu, ve kterém je potom patrná rotace kolem určitého středu. Dobře viditelný

případ je zachycen na obrázku 5.8 (délky vektorů jsou opět zvýrazněny). Z důvodu skládání pohybů navíc není střed rotace vždy uprostřed snímku. Pro popsání tohoto pohybu potřebujeme tento střed určit. Pokud budeme sledovat vertikální složku vektorů označenou d_Y v závislosti na jejich pozici v ose X, zjistíme, že se d_Y mění v závislosti na pozici na ose X a nabývá $d_Y = 0$ v bodě středu rotace. Obdobné platí pro složku vektorů d_X v závislosti na jejich pozici v ose Y. I když není tato závislost lineární, bylo ověřeno, že proložením závislosti d_Y na X přímkou $d_Y = k_1 x + k_0$ pomocí metody nejmenších čtverců (Least Square Method) lze získat souřadnici pravděpodobného středu rotace. Obdobně pak lze postupovat v ose Y pro výpočet zbývající souřadnice. Pro zjištění koeficientů k_0 a k_1 lze použít vzorec 5.3.

$$k_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}, k_0 = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2}. \tag{5.3}$$

Tato metoda se při testování ukázala jako dostatečně spolehlivá, vypočtený střed rotace je vidět např. na obrázku 5.9. Ve chvíli, kdy je střed známý, je možné zjistit úhel o který se pohled otočil a odečíst nechtěnou složku vektorů optického toku. Další možností je zachovat pouze tu složku vektorů, která směřuje ke středu rotace. V detekční metodě je použita druhá zmíněná možnost. Na obrázcích 5.8 a 5.9 je možné porovnat snímek před a po potlačení rotace.



Obrázek 5.8: Pohyb kamery způsobující rotaci v obrazu.

Obrázek 5.9: Tentýž snímek s vypočteným středem a potlačenou rotací.

5.5 Zpracování a sumace výsledků

Po vypořádání se s vedlejšími jevy pohybu kvadrokoptéry získáme délky vektorů rozmístěné po obrázku. Jejich směr není významný, pokud byly zachovány pouze určité složky podle popisu výše. Délky přesahující určitou hranici je možné považovat za blízkou překážku, neobvykle velké hodnoty pak obvykle představují chybu. Detekční metoda v bloku *Lokální navigace* ignoruje hodnoty pod hranicí `LOW_THRESHOLD` a nad hranicí `HIGH_THRESHOLD`. Délky jsou ohodnocovány lineárně od 0 až do 1 při délce `MEDIUM_THRESHOLD`. Větší hodnoty až do `HIGH_THRESHOLD` jsou stále ohodnoceny číslem 1.

Dosud popsaná metoda je již relativně použitelná, nicméně je nutné vypočtená ohodnocení dále zpracovávat v delším čase, než jednom samotném snímku. Nesprávně rozpoznané překážky se projevují jedním nebo malou skupinou vektorů v jediném snímku, přičemž v okolních snímcích chybí. Skutečná překážka se naopak vyskytuje postupně v několika

snímcích, a často je tvořena více jak jedním vektorem. Důležité je také vědět, kudy překážku obletět, obzvláště v případech, kdy je detekována slabě ohodnocená překážka na jedné straně a silně ohodnocená bližší překážka na druhé. K tomuto účelu slouží sumarizační mřížka o rozměrech 8×6. V každém snímku jsou všechny vektory rozděleny podle jejich pozice do buněk mřížky. V ní se uchovává počet vektorů a součet jejich ohodnocení.

Data ze sumarizační mřížky jsou uchovávána po dobu tří posledních snímků, celkové ohodnocení buňky v aktuálním čase je pak rovno průměru ohodnocení přes aktuální data a historii. Pro odstranění chyb je tento průměr snížen na třetinu, pokud je počet vektorů v buňce roven jedné.

5.6 Shrnutí

Kapitola popisuje prostředky použité při zpracování obrazu z kvadrokoptéry. Vhodnost práce v barevném prostoru HSV spočívá ve snadnější identifikaci jednoho odstínu barev v obraze. Toho se se využívá při detekci přistávací plošiny, která je identifikována na základě sytě červené barvy.

Detekce překážek je založena na optickém toku získaném metodou Lucas-Kanade. Z vektorů optického toku se snaží odečítat pohyby kamery, výsledné vektory pak ohodnocuje podle délky. Ohodnocení jsou sumarizována v rámci několika posledních snímků a na jejich základě je naplánován úhybný manévr. Rotace obrazu je potlačována za pomoci metody nejmenších čtverců pro získání středu rotace.

Kapitola 6

Implementace

Implementačním jazykem je C++. Pomocí objektového návrhu je možné zpřehlednit celý řídící systém, je ovšem nutné navázat tento uživatelský kód na aplikační kostru SDK, která je psána v jazyce C. Ke zpracování obrazu z kamer je využita knihovna OpenCV poskytující řadu algoritmů pro účely počítačového vidění. Knihovna je volně dostupná (pod licencí BSD), a to včetně dokumentace na [18]. K seznámení s OpenCV byla rovněž vydána kniha Learning OpenCV [4].

6.1 Formát obrazu

AR.Drone SDK dekóduje obrazové snímky až do rastrové podoby, kde jsou reprezentovány jako sled barev jednotlivých pixelů. Barevná informace jednoho pixelu má 24 bitů s rovnoměrným rozdělením 8 bitů na kanál. Hodnoty kanálů jsou poskládány ve sledu RGB, první byte snímku je tedy červená.

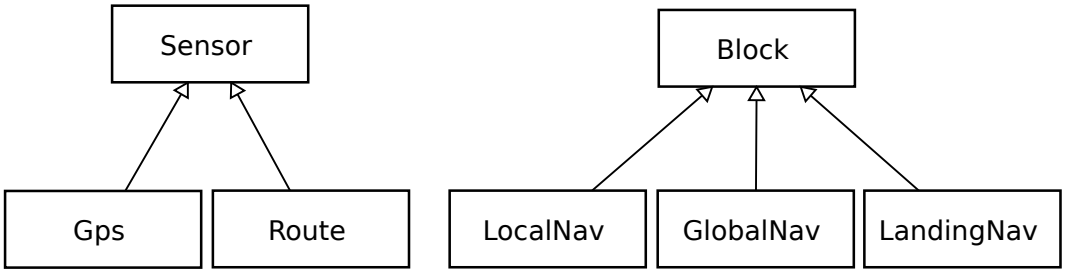
Reprezentace používaná v OpenCV je obdobná, barevné kanály jsou implicitně uloženy prokládaně [17]. Použitá barevná hloubka odpovídá `IPL_DEPTH_8U`, kanály jsou ovšem poskládány opačně ve sledu BGR, a tak je snímek potřeba konvertovat. Datová struktura obrázku použitá v OpenCV, `IplImage`, je v podstatě hlavička, která obrazová data pouze odkazuje prostřednictvím složky `imageData`. Vytvoření prázdné `IplImage` hlavičky a přepsání ukazatele na adresu snímku poskytnutého SDK umožňuje zpracování snímku knihovnou.

6.2 Koncept zdrojů dat

Klíčovou myšlenkou je umožnit experimentování se systémem "ze záznamu" mimo reálné prostředí, tedy zjišťování *co, proč a jak* se stalo, a upravování systému tak, aby se nad stejnými daty (tedy ve stejné situaci) choval korektně. Návrh počítá s reprezentací výše popsaných bloků navigační architektury třídami v C++, kde všechny senzory mají společné chování bázové třídy pro záznam dat a jeho následné čtení, a lze s nimi tak pracovat obecně (jak bude popsáno níže). Tyto třídy jsou pak využívány jak v *přehrávači* dat – aplikaci pro ladění, tak v řídicím programu propojeném s SDK. Vhodné je také zavést možnost automatického ukládání dat při reálném letu a výstup ladicích informací na konzoli.

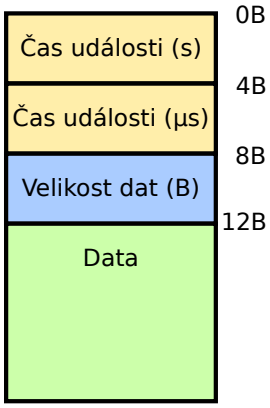
6.3 Třídy senzorů

Diagram tříd ukazuje obrázek 6.1. Bázová třída `Sensor` zachází s daty jako s bloky bajtů přijatými v určitou dobu – s časovým razítkem jejich příchodu. Po získání dat je automaticky zaznamená do souboru a potom předává všem výpočetním blokům, které se u ní zaregistrovaly.



Obrázek 6.1: Diagram tříd navržené řídicí aplikace.

Jak již bylo zmíněno v podkapitole 2.4, při příchodu dat v SDK je volána uživatelská funkce. Metodu pro příjem reálných dat je tedy nutné volat z této funkce. Při přehrávání záznamů pak stačí `Sensor` přepnout do režimu čtení ze souboru a místo toho volat funkci pro vyzvednutí další události ze souboru. K tomu účelu třída samostatně načítá hlavičku události s časem záznamu dopředu a umožňuje tak přehrávači řadit události správně za sebe.



Obrázek 6.2: Formát události v souboru se záznamem letu.

Navdata a video využívají přímo bázovou třídu `Sensor`, protože SDK předává jejich data vždy jako strukturu pevné velikosti. Rozlišení skutečného typu senzoru se provádí v konstruktoru, kde je specifikován identifikátor typu dat, který bude senzor oznamovat výpočetním blokům. Při každém příjmu dat se z kódu SDK volá metoda příjmu reálných dat třídy `Sensor`. Mezi parametry se předává ukazatel na data a jejich velikost.

GPS senzor používá svoji vlastní odvozenou třídu `Gps` s pevně nastaveným identifikátorem typu dat. Protože metoda příjmu dat nemůže v tomto případě fungovat jako předání ukazatele na blok paměti, je předefinována pro udržování TCP spojení s AR.Drone. Spojení navazuje stavový automat a udržuje jej v neblokujícím režimu, takže metoda může být volána periodicky (např. s příchodem navigačních dat) aniž by způsobila zamrznutí programu. V rámci příjmu NMEA dat tato třída porovnává kontrolní součty a dekóduje data do struktury, která je dále předávána již jako blok dat v paměti a tak je také ukládána na disk. V režimu čtení ze záznamu se již pracuje standardním způsobem.

Route je označován senzor pro trasu cesty a je reprezentován stejnojmennou třídou `Route`, opět s předefinovanou metodou pro získání dat. Místo bloku dat je jí předán řetězec znaků s názvem souboru, který má načíst. Každý řádek souboru představuje podcíl

trasy, který musí být zapsán ve formátu `DDMM.MMMM DDDMM.MMMM`, kde počet znaků určujících stupně je pevně daný (tak jako ve formátu věty `GPGGA`). Výsledkem čtení souboru je opět blok paměti, který obsahuje souřadnice trasy a je dále zpracováván.

Formát souboru se záznamem má jednoduchou strukturu podle obrázku 6.2. Každá událost má časové razítko ve formě POSIX time, tj. počtu sekund od 1. ledna 1970, a počet mikrosekund v rámci sekundy. Dále následuje informace o velikosti dat a samotná data. Tento způsob záznamu je sice velmi neúsporný u přijímaného videa, kde datový tok dosahuje 27000 Kib/s, nicméně nedochází ke ztrátě kvality obrazu a pro ladicí účely je tak plně vyhovující.

6.4 Třídy bloků

Bázová třída `Block` poskytuje metodu pro příjem dat ze třídy `Sensor`. Dále deklaruje metody pro vyvolání ladicího výstupu na konzoli a jeho přesměrování do souboru, a nastavení "simulačního" času v režimu čtení ze záznamu, kdy je čas nastavován přehrávačem.

Samotné SDK vypisuje na standardní výstup množství ladicích informací. Pro možnost sledovat větší množství vlastních výstupů, je deklarována virtuální metoda výstupu ladicích informací do souboru, kterou každý z bloků předefinovává a poskytuje ucelenou informaci o svém stavu. Místo souboru lze výstup přesměrovat do pojmenované roury (pipe) a posílat jej na terminál. Bloky takové použití předpokládají a posílají řídicí sekvence pro terminál, čímž formátují výstup. Takto je dosaženo jednoduchého oddělení více zdrojů informací, jelikož každý blok má svůj výstup a své okno terminálu.

Globální navigace je napojena na senzory souřadnic trasy a GPS. Příjem souřadnic trasy nastává typicky jednou, při startu řídicího programu, a vyvolává restart sledování cesty od prvního podcíle. K určení směru a vzdálenosti mezi aktuální pozicí a podcílem trasy se souřadnice přepočítávají do mřížky v kartézských souřadnicích. Nejdříve je určena vzdálenost mezi dvěma body v osách X a Y, z nich je následně vypočten azimut a vzdálenost k cíli. Chybu způsobenou zakřivením Země lze vzhledem k možnostem doletu kvadrokoptéry, který je kolem 5 km, zanedbat. Tento výpočet se spouští pouze na základě vyžádání od bloku *Lokální navigace*, kontroluje také kvalitu zaměřené pozice a vrací chybu v případě nízké přesnosti GPS nebo dlouhé doby od přijetí posledních dat.

K filtraci pozice je využito funkcí OpenCV `cvKalmanPredict()` a `cvKalmanCorrect()`. Mezi aktualizacemi je v matici přechodu F aktualizován čas dt od poslední korekce. Při příchodu nové hodnoty je pak přesnost nastavována podle hodnoty HDOP (Horizontal Dilution of Precision) udávané GPS modulem. Při dosažení podcíle je také zneplatněna předpovídaná rychlost pohybu, aby se potlačilo "přestřelení" pozice v prudkých zatáčkách. Tato filtrace byla implementována až dodatečně na základě testovacích letů.

Navigace na přistání zpracovává data z obou kamer a je aktivována *Lokální navigací* pouze v blízkosti cíle. Informace z bloku *Navigace na přistání* zahrnují pravdivostní hodnotu nalezení plošiny, její pozici a ohodnocení pro každou kameru zvlášť a také indikaci blízkosti plošiny na horizontální kameře. Princip detekce založený na syté barvě plošiny byl popsán v podkapitole 5.2 o zpracování obrazu.

Lokální navigace je blok řízený časovaným automatem fází letu, který byl popsán výše v podkapitole 4.2, a vypočítává hodnoty řízení, na které je periodicky dotazován z AR.Drone SDK. Jeho součástí je také zpracování hodnot z kompasu. Určení skutečné orientace kvadrokoptéry vzhledem k Zemi je klíčové obzvláště při natáčení do specifického směru, kdy je důležité mít k dispozici co neaktuálnější údaje, jinak by řízení bylo pomalé a těžkopádné.

Nejvyšší frekvence posílání dat z kompasu je 1 Hz. Proto blok *Lokální navigace* používá tuto hodnotu v kombinaci s gyroskopem, jehož hodnota přichází v navdatech. Ve chvíli, kdy jsou po startu řídicího programu dostupná data z kompasu, je vypočten rozdíl úhlů gyroskopu a kompasu, a skutečný směr kvadrokoptéry je pak dán součtem hodnoty z gyroskopu a této korekce. Tak je dosaženo velmi rychlé odezvy "kompasu" při zatáčení. Nutností je plynulé dorovnávání vypočtené korekce, gyroskop nemá totiž absolutní přesnost a jeho hodnota mírně "odplavává".

Další vlastností *Lokální navigace* je detekce překážek v obraze za letu a vyhýbání se jim. Metoda zpracování obrazu byla detailně popsána v podkapitole 5.3, jejím výsledkem je ohodnocení "nebezpečí" v různých oblastech obrazu. Z hlediska implementace využívá popsaná detekční metoda algoritmy knihovny OpenCV. Ta pro lokalizaci rohů nabízí funkci `cvGoodFeaturesToTrack()`, která umožňuje vynutit minimální "kvalitu" a vzdálenost dvou sousedních rohů. K výpočtu optického toku je použita varianta metody *Lucas-Kanade*, která pracuje s hierarchickým stromem, kde se obraz používá jak v původní velikosti, tak v několika menších, vždy polovičních velikostech. Výhodou je její schopnost vypořádat se s většími pohyby, které nelze vypočítat v původním rozlišení. Implementovaná metoda uloží první snímek video streamu a všechny další zpracovává následujícím způsobem.

- Ve starším snímku vyhledá rohy, které bude sledovat funkcí `cvGoodFeaturesToTrack()`.
- Tyto body pro větší citlivost detekce dohledá se sub-pixelovou přesností pomocí `cvFindCornerSubPix()`.
- Potom knihovní funkcí `cvCalcOpticalFlowPyrLK()` vypočítá optický tok.

Funkce výpočtu toku vrací kromě seznamu vektorů i míru chyby výpočtu, některé body se totiž nepodaří vůbec nalézt, nebo jsou nesprávné. Chybně vypočtené vektory, které se mezi výsledky běžně vyskytují jsou dále ignorovány.

6.5 Využití tříd

Navržené senzory a bloky navigační architektury jsou bez úpravy překládány *přehrávačem* i řídicí aplikací. Rozdílné nastavení v těchto aplikacích tedy musí zajistit zbylá část aplikace.

6.5.1 Přehrávač záznamů

Přehrávač je pomocná aplikace, kterou lze prohlížet data ze senzorů a krokovat chování navigační architektury. Z implementačního hlediska jde o funkci `main()`, která vytvoří instance objektů navigační architektury, propojí je podle schématu architektury a přepne zdroje dat do režimu čtení ze souboru. Po inicializaci ladicích výstupů vstoupí do přehrávací smyčky řízené uživatelem. K tomu využívá vstupu z klávesnice poskytovaného knihovnou OpenCV, která v rámci jednoduchých prvků uživatelského rozhraní, které obsahuje pro snadné experimentování s jejími algoritmy, umožňuje číst znaky stisknuté v některém z jejích oken. Pro podporu přehrávání událostí ve správném sledu umožňuje bázová třída `Sensor` zjistit

čas další události v souboru ještě před jejím úplným načtením a zpracováním. Přehrávač pak vybírá vždy nejbližší události a sekvenčně je provádí. Kromě přehrávání v reálném čase poskytuje možnost krokování po video snímcích i po jednotlivých událostech.

Při běhu algoritmů v přehrávači nastává problém s chováním závislým na reálném čase. Např. sledování aktuálnosti GPS dat při simulaci nesmí na rozdíl od reálného běhu číst systémový čas. Proto jsou zdroje dat vybaveny možností nastavovat "simulační" čas.

6.5.2 Řídicí program a interface s SDK

Přechod mezi řídicím programem a kódem SDK řeší sada funkcí v C++ překládaná tak, aby je šlo volat z kódu v C. Instance proměnných jsou vytvořeny a propojeny v inicializační funkci volané při startu programu. Dále se z SDK volají už jen obalovací funkce pro předávání dat. Jediná rozsáhlejší úprava kódu je nutná v místě čtení joysticku, kdy se navíc volá výpočet řízení z bloku *Lokální navigace*. To při zapnutém autonomním řízení nahrazuje manuální ovládání.

Z hlediska programovacího jazyka je pro umožnění volání C++ kódu z C zapotřebí deklarovat funkce s `extern "C"`, jak popisuje [16]. Pro více takto deklarovaných funkcí lze jejich deklarace umístit do bloku, před kterým je právě `extern "C"`.

6.6 Shrnutí

Podle rozvržení navigační architektury jsou implementovány třídy v jazyce C++, jejichž obecné pojetí senzoru usnadňuje vývoj podle záznamů letu. Dále byla popsána specifika odvozených tříd – třída senzoru `Gps` udržuje telnetové spojení s kvadrokoptérou a získaná data dál předává standardním způsobem, třída bloku *Lokální navigace* vyhlazuje hodnoty kompasu pomocí gyroskopu a dosahuje tak větší frekvenci aktualizace. Výsledkem implementace je aplikace přehrávače záznamů a samotná řídicí aplikace.

Kapitola 7

Testování

I když je možné odladit velkou část řízení ze záznamu, dynamické vlastnosti chování kvadrokoptéry jsou nezanedbatelné. Za letu se projevují jevy jako setrvačnost, vliv počasí, poruchy komunikace apod.

7.1 Chování AR.Drone při manuálním řízení

Co se týče chování AR.Drone, automatická stabilizace kvadrokoptéry za bezvětří se ukázala jako nad očekávání dobrá. AR.Drone se vyrovná s lehkým postrčením (pokusem o rozkývání) a vyrovnává náklon bez většího horizontálního pohybu. Jejího převrácení se (za bezvětří) docílit nepodařilo ani po silném impulzu, který ji naklonil o přibližně 45°. Během stabilizace z této polohy však již došlo ke ztrátě výšky – kvadrokoptéra dosedla z původní výšky 1,5 m na zem. Odolnost vůči otočení kolem svislé osy je značná, výkon vrtulí poskytuje relativně silný točivý moment na trup.

Mírně problematická je výšková stabilizace. Automatické udržování výšky může být nebezpečné v budovách, kde sonar zachycuje falešné odrazy, nebo po nalétnutí nad stůl způsobí náraz do stropu ve snaze vyrovnat původní výšku. Jisté problémy způsobuje i venku, kde je kvadrokoptéra sice schopna přeletět vegetaci jako keř či větší rostlinu, po jejím překonání má ale tendenci rychle snížit výšku na původní hodnotu, přičemž zadní rotory mnohdy těsně minou překážku a kvadrokoptéra navíc klesne níže než by měla. Tuto automatickou korekci nelze vypnout. Během letů se také několikrát objevily případy, kdy AR.Drone začala nekontrolovatelně stoupat i do výšky 10 m, i když to nikdy nebylo způsobeno ztrátou komunikace, kvadrokoptéra vždy zareagovala na požadavek automatického přistání.

Ve snaze zabránit takovému nebezpečnému stoupání bylo v řídicím programu doplněno nastavování maximální letové výšky kolem 3 m a provedena aktualizace firmware. Od té doby bylo pozorováno pouze jedno takto nebezpečné stoupání, nebylo však možné ověřit, za by se AR.Drone zastavila v maximální výšce, z důvodu kolize se stropem. Výška nad povrchem hlášená navdaty byla během tohoto stoupání 1,2 m, ve chvíli kolize přibližně 1,5 m.

7.1.1 Vliv větru

Za větru je jakákoli stabilizace znatelně horší. Vítr dosahující rychlosti 5 m/s a průměrné rychlosti 2,5 m/s (hlášený blízkou meteorologickou stanicí) způsobuje jak unášení kvadrokoptéry, tak její náhodné naklánění i otáčení kolem svislé osy. Kvadrokoptéra také svojí stabilizací ztrácí výšku.

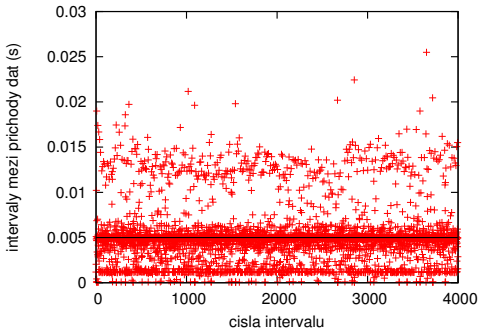
Obtížnějším problémem je rozkalibrování inerciální jednotky, pozorované při stálém bočním větru, kdy se na místě vznášející AR.Drone automaticky naklání proti větru, aby udržela nulovou horizontální rychlost. K problému dochází i při manuálně řízeném letu proti větru, kdy uživatel kompenzuje vítr. Výchylky joysticku přestanou po chvíli stačit a při jejich uvolnění do středové polohy se kvadrokoptéra prudce nakloní opačným směrem. Takové rychlé změny řízení za větru způsobují značný náklon kvadrokoptéry, několikrát bylo pozorováno i převrácení na střechu.

7.1.2 Bezpečnostní mechanismy

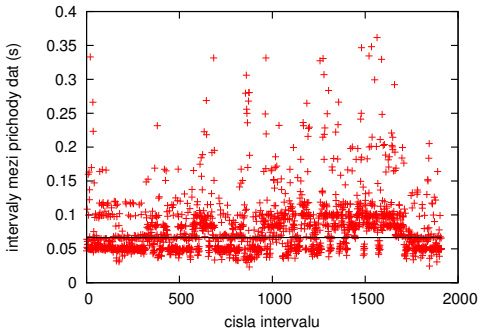
Při testování byla ověřena funkčnost řady bezpečnostních prvků AR.Drone. Silný vítr nebo nepřijatelně nekvalitní Wi-Fi spojení přepíná kvadrokoptéru do stavu zastavení `goto_fix` popsaného v článku [5]. Potvrzena byla i vlastnost vypnutí všech motorů při kontaktu vrtule s překážkou. K vypnutí postačil kontakt s volně visící čtvrtkou papíru. Do chybového stavu se přejde i při náklonu mimo rozsah $\pm 90°$. Bylo také ověřeno, že k horizontálnímu zastavení se používá vertikální kamery, firmwaru je údajně dostupných 60 snímku za sekundu. Za šera, nebo při přelepení kamery neprůhlednou páskou se AR.Drone nedokáže zcela zastavit na místě. Vnitřní senzory (akcelerometr) mohou poskytovat informaci pouze o zrychlení, takže rychlost relativně k zemi není známa.

7.1.3 Komunikace

Bezdrátový přenos není bezchybný, v průběhu letu se projevuje rušení a vyskytují se krátké výpadky spojení, které ztěžují řízení. Výpadky navigačních dat nebývají kritické, delší postřehnutelné intervaly jsou řešeny pozastavením řízení, např. u GPS, jehož rychlost reakce je v této aplikaci kritická. Znatelné poruchy se objevují ve videu, detekce překážek při nich musí být pozastavena, aby nedocházelo k chybnému rozpoznávání. Délky intervalů mezi příchody dat ukazují grafy na obrázcích 7.1 a 7.2 , navigační data jsou podle výrobce posílána v intervalech kratších 5 ms [12], snímková frekvence videa odpovídá intervalům 67 ms. Z grafů je vidět, že ke zdržení dat dochází zcela běžně.



Obrázek 7.1: Intervaly mezi příchody navigačních dat.

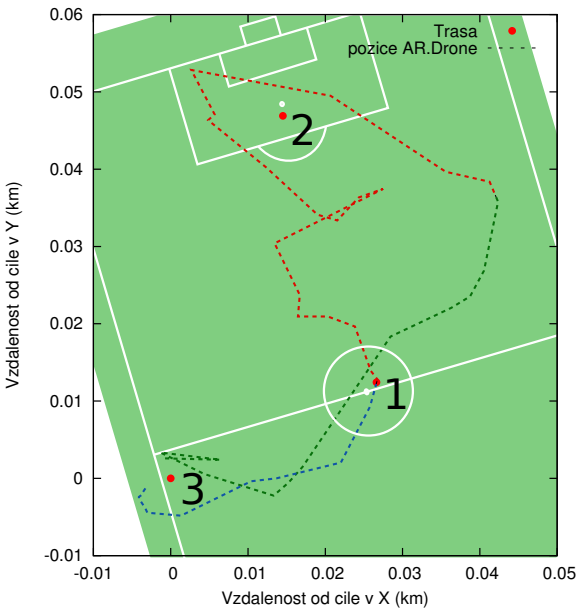


Obrázek 7.2: Intervaly mezi příchody obrazových snímků.

7.2 Autonomní řízení

Autonomní sledování trasy podle GPS v reálném prostředí funguje bez větších obtíží, během testování byla pouze zvýšena maximální povolená rychlost, aby bylo možné letět proti silnějšímu větru, a zavedeno zpomalování v blízkosti cílů. Kompas ve spojení s gyroskopem plynule natáčí kvadrokoptéru a nebylo třeba jeho chování upravovat. Chování během úspěšného testovacího letu je možné vidět na videozáznamu v datové příloze práce.

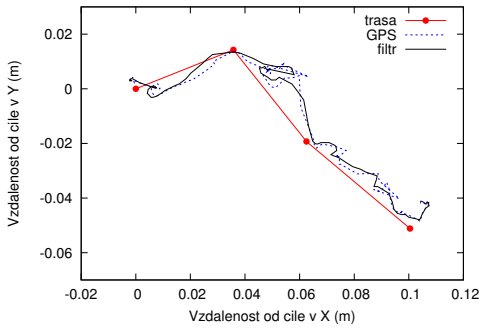
Nejslabším místem je nízká frekvence příchozích dat z GPS modulu, která ve spojení s nepřesnostmi polohy způsobuje nepříliš plynulé chování v blízkosti cílů. Naplánovanou trasu s průběžnou pozicí kvadrokoptéry hlášenou GPS reprezentuje obrázek 7.3, trasa začíná v bodě 3, postupuje body 1, 2 a končí v bodě 3 na přistávací plošině. Zaznamenaná trasa neodpovídá zcela skutečnému pohybu AR.Drone, pro porovnání je k dispozici již zmíněný videozáznam. Let kvadrokoptéry v obrázku začíná v bodě 3 a míří do středu hřiště (bodu 1), uprostřed spojnice bodů 1 a 2 je patrná chyba určení pozice vystřelením hlášené souřadnice na východ. Na videu se toto projevuje chybnou korekcí směru – přílišným natočením kvadrokoptéry doleva. V blízkosti bodu 2 pak došlo k chybnému zaměření pozice kvadrokoptéry v brankovišti, které způsobilo její pohyb na jihovýchod (v obrázku až na konec červené přerušované čáry), kde bylo nesprávně rozpoznáno dosažení bodu 2. Následný pohyb (zeleně) je již korektní ve směru cílového bodu 3 s mírnou korekcí směru vpravo v jeho blízkosti.



Obrázek 7.3: Záznam prolétnuté trasy podle GPS. (pozadí převzato z [19])

Aby se globální pozice zpřesnila, byl na vstupu dat do *Globální navigace* dodatečně implementován Kalmanův filtr (viz podkapitola 4.1). Odstranění šumu je ilustrováno na obrázku 7.4, menší výkyvy pozice jsou eliminovány, významnější chyby jsou ale promítnuty do výstupu. Ve videu v datové příloze práce je filtrovaná pozice značena zeleným křížkem.

Během testování se také ukázalo, že vyhledávání plošiny v cíli je náchylné vůči větru, během rozhlédnutí totiž dochází k odfouknutí kvadrokoptéry mimo viditelnost plošiny. Obdobný problém představuje dosedání na $60 \times 60\,cm$ velkou vyvýšenou plošinu, pokud není úplné bezvětří. Aby se tomu předešlo, byla fáze `LOOK_AROUND` upravena tak, aby se rychleji ukončila v blízkosti plošiny, přistávací manévr je při ztrátě plošiny ukončován až po uplynutí určeného času, během kterého jsou udržovány poslední vypočtené hodnoty řízení. Lineární křivka rychlosti v závislosti na pozici plošiny byla také nahrazena křivkami popsanými v podkapitole 4.4.



Obrázek 7.4: Trasa letu, pozice podle GPS a pozice filtrovaná Kalmanovým filtrem.

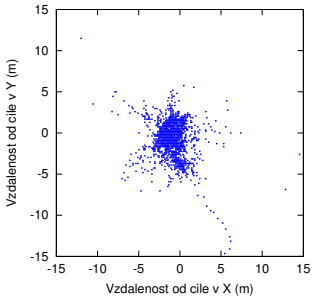
Obrázek 7.5: Obraz ze spodní kamery (vlevo nahoře) zabírající červenou přistávací plošinu.

7.2.1 Přesnost GPS

Přesnost modulu LS20126 je podle katalogového listu ±5 m [9]. V otevřeném prostranství s dobrým výhledem na oblohu byl proveden pokus, kdy byla zaznamenávána poloha hlášená modulem po dobu jedné hodiny, zatímco byl položen ve výšce 25 cm nad zemí. Přijímači byla umožněna inicializace a zaměření platné pozice, při začátku měření bylo sledováno 6 satelitů a udávána přesnost ±10,5 m. Během pokusu se počet satelitů dostal až na 9, udávaná přesnost dosahovala až ±4.5 m, kolísala však a zřídka se dostávala i pod ±10 m. "Trasa" GPS, která je výsledkem měření je vidět na obrázku 7.6. Vypočtená směrodatná odchylka je 2,475 m, více než 95% naměřených hodnot ze získaných dat tedy leží ve vzdálenosti menší než 4,951 m od průměrné hodnoty. K porovnání absolutní přesnosti by bylo nutné znát velmi přesnou pozici bodu, ve kterém byl modul umístěn.

7.2.2 Sledování přistávací plošiny kamerami

Problematickým senzorem pro účely řízení počítačem, je vertikální kamera, resp. její snímání barvy, kvůli kterému bylo nutné několikrát překonfigurovávat rozsahy barev pro detekci přistávací plošiny. Plošina sytě červené barvy je na obrázku 7.5 zbarvena do růžova až fialova. V přiloženém videu letu je možné pozorovat řadu neúspěšných pokusů o přistání na konci trasy. To bylo způsobeno právě touto chybou, barva plošiny byla za jasného světla rozpoznána až při posledním pokusu. Obě použité kamery také za neideálních světelných podmínek trpí znatelným šumem v obraze. Ten má obvykle značnou sytost barvy a prochází tak prahováním do výsledného obrazu. Tento jev umožnilo redukovat tzv. erodování snímku. Metoda ponechá v obrázku pouze ty pixely, které mají ve svém okolí dostatečný počet nenulových (světlých) pixelů. Osamostatněné pixely, které jsou typicky způsobené šumem jsou odstraněny.



Obrázek 7.6: Pozice hlášená stojícím GPS přijímačem v průběhu jedné hodiny.

7.2.3 Detekce překážek

Kvadrokoptéra je schopna se za letu vyhnout některým statickým překážkám a většině těch pohyblivých. Největší spolehlivost rozpoznání překážky je na okraji obrazu, kde bývá zjišťován největší optický tok. Překážky přímo před kvadrokoptérou, stejně jako jednobarevné plochy (na kterých se nevyskytují rohy vhodné pro sledování) jsou naopak obtížně rozpoznatelné. Taktéž překážky jako tyče a dlouhé tenké kmeny vysazených stromků neposkytují dostatek vhodných rohů, jak již bylo naznačeno v podkapitole 5.3 (viz *Aperture problem*). Příklad detekované překážky za letu je vidět na obrázku 7.7, v datové příloze práce je rovněž k dispozici videozáznam úhybných manévrů. Použitá detekční metoda byla nakalibrována na zachycování i drobnějších překážek, dochází tedy i k řadě chybných detekcí, nicméně korekce směru pohybu je u nich mírná a neovlivňuje celkový směr letu. Chybně rozpoznaná překážka (způsobená příliš velkým optickým tokem v okolním terénu) je vidět na obrázku 7.8.

Obrázek 7.7: Správně rozpoznaná překážka za letu.

Obrázek 7.8: Chybně rozpoznaná překážka.

7.3 Kalibrace parametrů

Implementovaný program obsahuje řadu parametrů pro přizpůsobení se terénu, ve kterém bude nasazena. Tyto parametry musí být vhodně nastaveny pro jeho správnou funkčnost. Hodnoty i s popisem jejich funkčnosti jsou obsažené v hlavičkových souborech bloků navigační architektury.

`localnav.h` obsahuje nastavení bloku *Lokální navigace*. Předně je zde nastavován rozsah hodnot řízení, časování fází letu a citlivost detekce překážek.

- Maximální náklon kvadrokoptéry byl nastaven na 11,5° (0,2 rad), tato hodnota poskytuje subjektivně dostačující výchylky pro manuální ovládání, řídicí program pak využívá jen část této hodnoty. Rychlost při sledování trasy je nastavena na 30%, maximální rychlost navádění na plošinu je 10%. Úhybné manévry přičítají k aktuálním hodnotám řízení nejvýše 20% maximální rychlosti. Rychlost otáčení (nezávislá na úhlu náklonu) dosahuje nejvýše 80%.
- V rámci řízení je také definována letová výška, které je přizpůsobeno nastavení detekce překážek, let příliš nízko by totiž znamenal zachytávání překážek na zemi. Optimální

výška je stanovena na 1 m. Další sledovaná výška je ta, na které se při dosedání na plošinu vydá přistávací povel pro SDK. Ta byla zvýšena z původních 40 cm na 60 cm.

- Dále jsou nastaveny parametry fází letu v blízkosti plošiny. Délky trvání fáze přiblížení na plošinu `SCAN_FORWARD` je vyhovující při 5 s, dvě opakování s fází `LOOK_AROUND` jsou plně dostačující. Délka intervalu před přerušením přistávacího manévru byla zvýšena na 3 s, to zvyšuje šanci znovunalezení plošiny po jejím zmizení z obrazu vertikální kamery.

- Hodnoty týkající se detekce překážek nastavují velikost obrazu 320×240 px a jeho ignorovanou spodní část, kde se vyskytují chybně vypočtené vektory na zemi. Nastavená hodnota je 60 px (čtvrtina obrazu). Potřebný výpočetní výkon řídicího počítače je závislý na počtu sledovaných rohů v obraze při výpočtu optického toku. Tato hodnota byla experimentálně nastavena na 128. Při větších hodnotách se stávalo, že v některých scénách výpočet nestíhal. Nastavení minimální vzdálenosti sousedních rohů a jejich minimální kvalita byly zvoleny podle literatury [4]. Z hlediska řízení je možné nastavit minimální délku úhybného manévru, hodnota vyhovující i malým překážkám je 0,5 s. Nejcitlivější částí je nastavení ohodnocení délek vektorů. Ignorovány jsou délky pod 4 px a nad 24 px, 100% ohodnocení je dosaženo již na délce 8 px. Toto nastavení detekuje i menší překážky, jak již bylo popsáno v podkapitole 7.2.3.

`landingnav.h` určuje parametry *Navigace na přistání* a spočívá ve správném nastavení intervalů hodnot HSV pro rozpoznání plošiny. Tyto intervaly byly nastaveny tak, aby vyhovovaly světelným podmínkám ve všech záznamech, které byly během testování pořízeny. Vertikální kameře musel být definován vlastní interval barevného odstínu kvůli nestálosti její barvy. Obdobně byly nastavovány i počty detekovaných pixelů, které jsou již považovány za plošinu.

- Interval sytosti barvy byl nastaven na $\langle 90; 255 \rangle$, světelnost $\langle 110; 255 \rangle$.

- Barevný odstín pro horizontální kameru ve stupních je $\langle 336; 358 \rangle$, pro vertikální kameru musel být rozšířen na $\langle 290; 358 \rangle$.

`globalnav.h` pro *Globální navigaci* nastavuje pouze vzdálenost, ve které je cíl považován za dosažený a obvody Země kolem rovníku a přes póly. Vhodná vzdálenost dosažení cíle je 7 m, při nižších hodnotách se kvadrokoptéra chová zmateně a dlouho létá kolem cílového bodu, zatímco následuje pouze nepřesnost GPS.

7.4 Shrnutí

Chování AR.Drone je silně závislé na větru. Autonomní řízení je funkční, je schopné navigovat kvadrokoptéru na cílový bod, kde přistane na plošině. Kvůli nižší rychlosti aktualizace a přesnosti GPS modulu se při sledování trasy projevují menší výkyvy v trase. Přistávací fáze je snadno ovlivněna větrem, nicméně byla úspěšně otestována. Vyhýbání se překážkám je taktéž funkční, má však problémy s překážkami, které nezpůsobují optický tok v obraze.

Kapitola 8

Závěr

Autonomní navigace různých strojů ve venkovním prostředí se díky dnešní dostupnosti globálních navigačních systémů stala možnou a běžně dostupnou. V této práci byl popsán návrh autonomního řízení kvadrokoptéry AR.Drone, která je primárně určena pro hrání her s obohacenou realitou, jednoduchost jejího ovládání však nabízí možnost jejího použití pro experimentální účely. Její senzorický systém byl rozšířen o senzor GPS a magnetický kompas. Tyto senzory byly připojeny přímo k hardwaru kvadrokoptéry, čímž bylo umožněno přenášení dat po existujícím Wi-Fi spojení.

Navržená navigační architektura obsahuje bloky zodpovědné za globální a lokální navigaci a za detekci přistávací plošiny. Globální navigace spočívá ve výpočtu směru k dalšímu podcíli, který je preferován lokální navigací. Vyhledávání přistávací plošiny se spoléhá na syté barevné označení plošiny. Lokální navigace je pak řízena časovaným automatem, podle kterého řídí kvadrokoptéru buď ve směru podcílů trasy, anebo hledá přistávací plošinu v okolí.

Řídicí program je v reálném prostředí schopen navigovat AR.Drone po trase na cílové souřadnice. Při tom se spoléhá na kombinaci GPS, kompasu a gyroskopu k určení své polohy a směru k cíli. Autonomní vyhledání a přistání na plošině v cíli je taktéž funkční, nicméně je za přítomnosti větru méně spolehlivé, protože kvadrokoptéra je větrem silně ovlivňována. V rámci vývoje řídicího programu byl také navržen přehrávač letových záznamů, který umožňuje offline experimentování s řídicím systémem.

Dále byla navržena a otestována detekce překážek za letu. Ta pracuje na základě výpočtu vektorů optického toku, ze kterých odečítá nežádoucí pohyby kamery způsobené manévrováním kvadrokoptéry. Metoda rozpoznává překážky podle nadlimitní délky těchto vektorů způsobených blízkými objekty. Během letu je tak kvadrokoptéra schopna vyhýbat se chodcům a některým statickým překážkám.

Článek shrnující principy popsané v této práci byl přijat do soutěže STUDENT EEICT 2012 pořádané Fakultou elektrotechniky a komunikačních technologií a Fakultou informačních technologií VUT v Brně, kde získal 1. místo v sekci Inteligentní systémy bakalářské formy studia.

8.1 Možnosti rozšíření

V dalším vývoji bude možné pokračovat v řadě oblastí. Vylepšení spolehlivosti detekce překážek by bylo možné dosáhnout kombinací existujícího algoritmu se sledováním jednobarevných ploch, které nejsou optickým tokem rozpoznatelné. Z hlediska potlačení nežádoucích

pohybů kvadrokoptéry by bylo možné nastavovat citlivost detekce adaptivně v závislosti na zrychlení stroje vyžádaném řídicím programem.

V oblasti hledání přistávací plošiny se nabízí sledovat pohyb země na vertikální kameře a odhadovat tak aktuální vzdálenost kvadrokoptéry od plošiny, pokud byla již jednou nalezena. To by pomohlo k lepšímu odolávání větru v této fázi letu. Ideální vzhledem k rychlosti odezvy řízení by bylo pokusit se přesunout méně náročnou část řízení do vestavěného počítače AR.Drone, takové zásahy však nejsou výrobcem podporovány.

Na desce plošných spojů s napájením pro GPS modul je možné osadit napěťový stabilizátor ve větším pouzdře. Ač je dosahovaná teplota bezpečně v rámci pracovního rozmezí, aktuální pouzdro SO-8 se při provozu zbytečně zahřívá. Důvodem je napěťová ztráta kolem 7,8 V při proudu 33 mA, a maximální modulem odebíraný proud při inicializaci, který byl ve starších verzích katalogového listu uváděn nižší.

Literatura

[1] ALUR Rajeev a David L. Dill: A theory of timed automata [online]. *Theoretical Computer Science 126*, 1994, s. 183-235 [cit. 2012-04-23], Dostupné z: `http://www.cmi.ac.in/~madhavan/courses/verification-2011/alur-dill-tcs-94.pdf`.

[2] BOUCHAFA, Samia a Bertrand Zavidovique: c-Velocity: A Flow-Cumulating Uncalibrated Approach for 3D Plane Detection [online]. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, Apr 2012, roč. 97, č. 2, str. 148-166. [cit. 2012-04-20], Dostupné z: `http://www.springerlink.com/content/f3845610m2185442/fulltext.pdf`.

[3] BOUCHAFA, Samia a Bertrand Zavidovique: Obstacle Detection "for free" in the C-velocity space [online]. *14th International IEEE Conference on Intelligent Transportation Systems*, OCT 05-07, 2011, str. 308-313. [cit. 2012-04-20], Dostupné z: `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=06082872`.

[4] BRADSKI, Gary a Adrian KAEHLER: *Learning OpenCV: Computer Vision With the OpenCV Library*. Sebastopol: O'Reilly, 2008, ISBN 9780596516130.

[5] BRISTEAU, Pierre-Jean et al.: The Navigation and Control technology inside the AR.Drone micro UAV [online]. *Preprints of the 18th IFAC World Congress Milano (Italy)*, August 28 - September 2, 2011, s. 1477-1484. [cit. 2012-04-17], Dostupné z: `http://cas.ensmp.fr/~petit/papers/ifac11//PJB.pdf`.

[6] COOPER, Bills, Chen Joyce a Saxena Ashutosh: Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues [online]. *International Conference on Robotics and Automation (ICRA), 2011*, 2011 [cit. 2012-04-20], Dostupné z: `http://www.cs.cornell.edu/~asaxena/MAV/saxena_MAV_perspectivecues_stairs.pdf`.

[7] ERESEN, Aydin, Nevrez Imamoglu, Mehmet Onder Efe c: Autonomous quadrotor flight with vision-based obstacle avoidance in virtual environment [online]. *Expert Systems with Applications 39*, 2012, s. 894-905. [cit. 2012-04-20], Dostupné z: `http://cas.ensmp.fr/~petit/papers/ifac11//PJB.pdf`.

[8] LOCOSYS Technology Inc.: Core Module Calibration Procedures [online]. 2006 [cit. 2012-04-17], Dostupné z: `http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/core%20module%20calibration%20procedures%20.pdf`.

[9] LOCOSYS Technology Inc.: Datasheet of stand-alone GPS smart antenna module with magnetic sensor, LS20126 [online]. 2010 [cit. 2012-04-17], Dostupné z: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/GPS/LS20126_datasheet_V1.02.pdf.

[10] LOCOSYS Technology Inc.: Locosys Technology [online]. [cit. 2012-04-17], Dostupné z: http://www.locosystech.com/product.php?id=57.

[11] National Marine Electronics Association: NMEA 0183 Standard [online]. [cit. 2012-04-17], Dostupné z: http://www.nmea.org/content/nmea_standards/nmea_083_v_400.asp.

[12] Parrot SA: AR.Drone Developer Guide [online]. 2011 [cit. 2012-04-17], Dostupné z: https://projects.ardrone.org/attachments/download/365/ARDrone_SDK_1_7_Developer_Guide.pdf.

[13] Parrot SA: ARDRONE open API platform [online]. [cit. 2012-04-17], Dostupné z: https://projects.ardrone.org/.

[14] Parrot SA: AR.Drone Parrot - The flying video game [online]. [cit. 2012-04-17], Dostupné z: http://ardrone.parrot.com/parrot-ar-drone/en/.

[15] Tech Toy Hacks: AR.Drone Parrot [online]. 2011 [cit. 2012-04-17], Dostupné z: http://blog.perquin.com/blog/category/ardrone/.

[16] VIRIUS, Miroslav: *Jazyky C a C++: kompletní kapesní průvodce programátora*. 1. vyd., Praha: Grada, 2006, 518 s, ISBN 80-247-1494-9.

[17] WWW stránky: *OpenCV 2.0 C Reference: Basic Structures [online]*. [cit. 2012-04-17], Dostupné z: http://opencv.willowgarage.com/documentation/basic_structures.html?highlight=imageData#iplimage.

[18] WWW stránky: OpenCV Wiki [online]. [cit. 2012-04-17], Dostupné z: http://opencv.willowgarage.com/wiki/.

[19] WWW stránky: Wikimedia Commons [online]. [cit. 2012-04-17], Dostupné z: http://commons.wikimedia.org/wiki/Hlavní_strana.

Seznam příloh

Příloha A

Obsah CD

Datová příloha práce obsažená na CD obsahuje:

- Přeloženou podobu práce a její verzi pro tisk.
- Složku `tex` se zdrojovými soubory práce pro LaTeXvčetně obrázků.
- Příspěvek "Quadrocopter Navigation and Control" přijatý do soutěže STUDENT EEICT 2012.
- Videa z několika zkušebních letů včetně letu popisovaného v podkapitole 7.2.
- Tištěný spoj se stabilizátorem napětí pro GPS modul navržený v programu EAGLE.
- Samotný řídicí program. Jeho překlad vyžaduje přítomnost knihovny ARDroneLib v systému souborů na stejné úrovni.
- Složku `ARDroneLib` se soubory potřebnými k překladu řídicí aplikace.
- Přehrávač záznamů se skripty spouštějícími ukázkové záznamy letu.
- Knihovnu OpenCV nutnou k překladu a spuštění obou programů.
- Katalogové listy zvoleného GPS modulu.

Příloha B

Postup spuštění řídicího programu

Implementovaný řídicí program běží pod operačním systém GNU/Linux, jeho překlad vyžaduje systém s nainstalovanou knihovnou OpenCV. Ke spuštění zkušebního letu je potřeba postupovat následujícím způsobem:

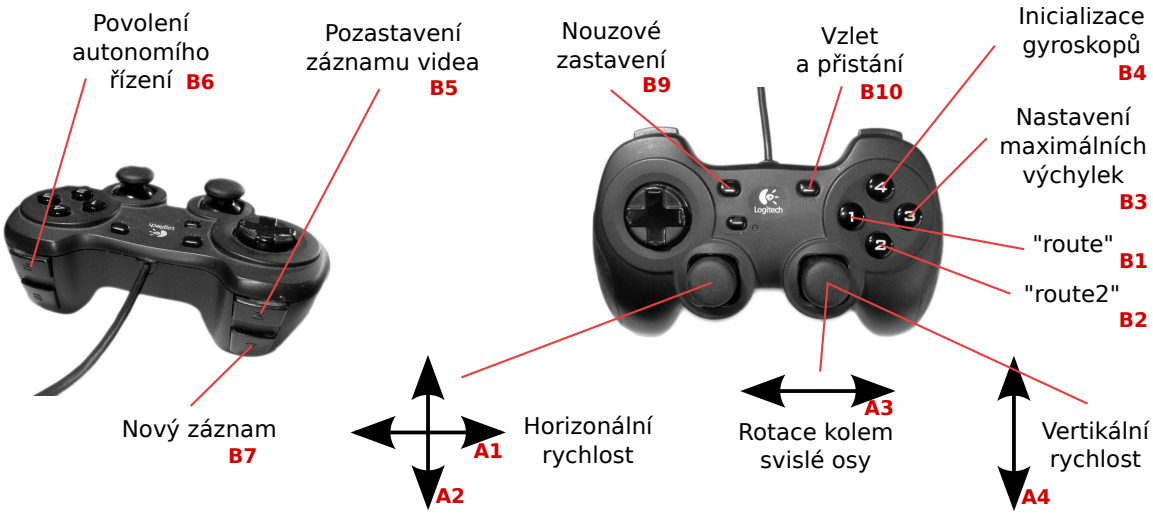
- Vytvořit soubor `route` s trasou. Formát souboru, popsaný v 6.3, musí být dodržen.
- V AR.Drone SDK zkontrolovat USB identifikaci VID a PID typu použitého joysticku. Pokud se identifikace liší od přednastavené hodnoty `046d:c216`, je nutné ji změnit v souboru *UI/gamepad.h* (hodnota `GAMEPAD_LOGICTECH_ID`) a poté přeložit aplikaci.
- V systému je nutné zkalibrovat joystick tak, aby byly výchylky ve všech osách stabilně na nule. To je obzvlášť důležité, neboť pohyb mimo nulovou hodnotu je považován za manuální převzetí řízení a tedy odpojení autonomního ovládání. Popis kalibrace joysticku je popsán na konci přílohy.
- Pokud byl řídicí program přeložen se zapnutými ladicími výstupy, je nutné spustit terminály, které je čtou.
- Umístit AR.Drone na rovný povrch kvůli inicializaci gyroskopů a zapojit napájecí akumulátor.
- Na řídicím počítači se připojit na přístupový bod `ardrone_NNNNNN` a spustit řídicí aplikaci.
- Nyní by již AR.Drone měla reagovat na manuální řízení. Rozvržení ovládacích prvků je popsáno níže.

V případě, že není nastaveno automatické spouštění firmwaru, nebo AR.Drone nekomunikuje, je potřeba připojit se přes telnet na `192.168.1.1` a restartovat program `/bin/program.elf`

B.1 Popis ovládání

Manuální ovládání je přizpůsobeno běžnému typu gamepadů a je rozvrženo podle obrázku B.1. Joystick nalevo řídí horizontální pohyb kvadrokoptéry – pohyb vpřed/vzad a "úkroky stranou" vlevo/vpravo. Pravý joystick řídí otáčení kolem svislé osy a stoupání/klesání. Vzlet a přistání jsou spouštěny tlačítkem 10, které je na mnoha ovladačích často označeno jako

start, je tak nazýváno i v kódu SDK. Tlačítko 9, často značené *select*, způsobuje, že je vysílán tzv. *emergency signal*, který přepíná řídicí firmware do *default* stavu, tzn. okamžitě vypne motory a způsobí tvrdý pád na zem. Pokud byl program přeložen se zapnutými ladicími nástroji, ovládají tlačítka 5 a 7 nahrávání senzorových dat. Tlačítko 5 při podržení pozastavuje záznam příchozího videa, stisk tlačítka 7 způsobí uložení aktuálního záznamu a začne nahrávat do nového souboru.



Obrázek B.1: Ovládání řídicího programu gamepadem.

Autonomní řízení je povolováno držením tlačítka 6. V případě potřeby je možné uvolněním tlačítka nebo pohybem některého z joysticků převzít manuální řízení. Pokud bylo převzato manuální řízení, je pro opětovné povolení autonomního chování nutné tlačítko 6 uvolnit a znovu stisknout. Znovunačtení souborů s trasou `route` resp. `route2` způsobuje stisk tlačítek 1 resp. 2.

B.2 Kalibrace joysticku

Po připojení joysticku je v systému vytvořeno zařízení `/dev/input/js0`, případně s vyšším číslem, pokud je jich v systému více (zde zmíněné příkazy je pak nutné upravit). Funkčnost lze otestovat pomocí programu `jstest` z balíku `joystick`, příkazem `jstest /dev/input/js0`.

Programem zobrazené hodnoty by měly být ve všech osách nulové, je-li joystick v klidu. Pokud tomu tak není, existuje pro kalibraci program `jscal` ze stejného balíku, který se spouští příkazem `jscal -c /dev/input/js0`. Naneštěstí automatická kalibrační procedura v použité verzi nenastavuje na osách "mrtvé zóny" kolem nuly, takže bylo nutné ruční zadání koeficientů.

Pokud není po automatické kalibraci chování os v pořádku, je možné kalibrační parametry zadávat ručně pomocí přepínače `-s`. Parametry jsou tvořeny číselnými hodnotami oddělenými čárkou, kde první hodnota udává počet os n, a za ní následuje n krát nastavení osy. Každé nastavení osy tvoří šestice hodnot: `1,0,`$dead_{MIN}$`,`$dead_{MAX}$`,`$factor_{-}$`,`$factor_{+}$, kde $dead_{MIN}$ a $dead_{MAX}$ určují interval, kde má být osa v nulové pozici. $factor_{\pm}$ je číslo, kterým se násobí zbylá výchylka osy, aby se v jejím maximu dosáhlo hodnoty 32767. Při tom se uvažuje, že syrová data jsou v rozsahu 0 až 255, tak jak je v průběhu kalibrace

zobrazuje `jscal`. Z nezjištěného důvodu jsou čísla $factor_{\pm}$ při zadávání na příkazové řádce dodatečně vynásobena hodnotou 16384.

Příklad postupu ruční kalibrace je následující. Pro čtení syrových dat se využije pomocí programu `jscal`. První osa např. zobrazuje hodnoty od 0 do 255. Páčka se v nulové pozici mírně viklá mezi hodnotami 123 a 139, což jsou přímo hodnoty $dead_{MIN}$ a $dead_{MAX}$. Zbývající rozsah os po odečtení "mrtvého" intervalu je 123 vlevo a $255 - 139 = 116$ vpravo. Z toho plynou hodnoty $factor_{-} = 32768/123$ a $factor_{+} = 32767/116$. Hodnoty $factor_{\pm}$ jsou ještě vynásobeny 16384, šestice hodnot pro kalibraci první osy tak vypadá takto: `1,0,123,139,4364804,4628197,`. Příkaz příkazové řádky vypadá například takto:

```
jscal -s 6,1,0,123,139,4364804,4628197,1,0,123,136,4364804,4511520,
         1,0,128,147,4194304,4971026,1,0,113,134,4751070,4436949,
         1,0,0,0,536870912,536870912,1,0,0,0,536870912,536870912
         /dev/input/js0
```

Příloha C

Diplom ze soutěže EEICT 2012

Vysoké učení technické v Brně
Fakulta elektrotechniky a komunikačních technologií
Fakulta informačních technologií

FIT

Konference a soutěž studentské tvůrčí činnosti

STUDENT EEICT

V Brně dne 26. dubna 2012

Diplom

KAREL DOLEŽAL

z Fakulty informačních technologií

za 1. místo v sekci

Inteligentní systémy
bakalářské formy studia

Konference a soutěže studentské tvůrčí činnosti
STUDENT EEICT 2012

Hodně úspěchů ve Vašich dalších odborných aktivitách.

Prof. RNDr. Vladimír Aubrecht, CSc.
Předseda Rady EEICT