Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra Počítačů

Diplomová práce

Ovládání týmu robotických prostředků s využitím augmentované reality

Bc. Ondřej Záruba

Vedoucí práce: Ing. Milan Rollo, Ph.D.

7. ledna 2016

Poděkování

Děkuji Ing. Milanovi Rollovi, Ph.D. za odborné vedení této práce a také všem studentům, kteří tuto práci testovali a přispěli cennými podněty.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 7. ledna 2016 .....

Odkaz na tuto práci

Záruba, Ondřej. *Ovládání týmu robotických prostředků s využitím augmentované reality*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2016.

Abstrakt

Ovládání týmu bezpilotních prostředků klade zvýšené nároky na lidského operátora. V rámci této práce byly vyvinuty metody pro snížení jeho kognitivní zátěže s využitím augmentované reality. Ta umožňuje přehlednější zobrazení pozic jednotlivých prostředků v prostoru, vizualizaci jejich stavu a aktuálních úkolů. Byla navržena a implementována aplikace, prostřednictvím které může operátor zadávat bezpilotním prostředkům úkoly a v případě potřeby je manuálně dálkově ovládat. Aplikace byla vytvořena s důrazem na minimalizaci ceny potřebného hardware. Výsledná implementace byla otestována na virtuálním i fyzickém bezpilotním prostředku a byla zhodnocena její funkčnost.

Klíčová slova UAS, augmetovaná realita, virtuální realita, dálkové ovládání

Abstract

Control of a team of unmanned vehicles lays an increased pressure on a human operator. Methods to lower the cognitive pressure through the use of augmented reality have been developed within this thesis. This technology allows synoptic representation of vehicles in space, visualization of their state and current tasks. New application which allows the operator to submit tasks to the vehicle and manually overtake its control in case of need was designed and developed. This application was developed with focus on minimizing the price of the hardware required. Resulting implementation was tested on virtual as well as physical unmanned vehicle and its performance was evaluated.

Keywords UAS, augmented reality, virtual reality, remote control

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Poslední dobou zažívá rozkvět používání bezpilotních prostředků. Ještě v nedávné době byla taková zařízení doménou laboratoří a armády, ale díky výrazně se snižující ceně si je dnes může dovolit prakticky každý. Díky tomuto přichází nové možnosti uplatnění v civilním sektoru. Nízká cena také umožňuje stavbu levných specializovaných robotů, kteří spolu kooperují pro dosažení společného cíle. Přestože cílem je vytvářet bezpilotní prostředky autonomní, je pravděpodobné, že ještě dlouho bude v průběhu plnění zadaných úloh nutný zásah operátora. Zásahy jsou to většinou normální jako například plánování tras letu nebo zadávání úkolů, ale mohou zahrnovat i řešení neočekávaných situací nebo úkolů příliš složitých pro naprogramování. Pro usnadnění zásahů operátora do plnění úloh, případně pro ovládání celého týmu bezpilotních prostředků je třeba hledat nové způsoby a metody interakce. Jednou z možností je využití augmentované reality.

Mezi lety 2012 až 2014 došlo k představení a uvedení na trh několika spotřebitelských zařízení, jež umožnily vytvořit tuto práci. Prvním zařízením byly brýle pro zobrazování virtuální reality jménem Oculus Rift. Oculus ukázal, že cena hardware, kvalita zobrazovací techniky a poziční technologie se dostaly na takovou úroveň, aby virtuální realita vystoupila z laboratoří a průmyslu mezi běžné uživatele. Dalším zařízením jsou Google Glass, jež představují první krok k brýlím s augmetovanou realitou pro běžné použití. Posledním je pak, z kartonového papíru a plastových čoček vyrobený, Google Garboard, což je extrémně levná jednoduchá skládačka brýlí pro virtuální realitu, k jejímuž oživení stačí dnes již i průměrný mobilní telefon.

Tato práce popisuje systém, jenž se snaží snížit zátěž operátora s využitím augmentované reality. Systém se skládá ze softwaru pro operátora a bezpilotní prostředky. Práce také obsahuje doporučený hardware, na kterém byly provedeny pokusy pro ověření funkčnosti navrhovaného řešení.

Kapitola 1

Přehled dostupných technologií

1.1 Bezpilotní prostředky

Bezpilotní prostředky jsou prostředky, které nevyžadují pro svou činnost člověka na palubě a jsou řízeny na dálku nebo pomocí autonomních systémů. Těchto zařízení existuje obrovské množství a operují prakticky kdekoliv od mořských hlubin až do vzdálených částí naší sluneční soustavy. Bezpilotní prostředky lze rozdělit podle prostředí výskytu dle tabulky 1.1 nebo podle stupně samostatnosti řízení. Řízení lze rozdělit mezi ruční, asistované a autonomní.

Hlavním hybatelem v oblasti bezpilotních prostředků byla a je armáda. Proto není divu, že první bezpilotní prostředky byly létající bomby a torpéda. Za jednoho z prvních předchůdců moderních UAV se dá považovat americký „The Charles Kettering Aerial Torpedo“ z roku 1918, který používal k řízení mechanický gyroskop a barometr. K detekci dosažení cíle byl použit čítač otáček motoru, jenž po dosažení nastavené hodnoty vypnul motor a zasunul západky držící křídla. K výpočtu počtu otáček motoru nutných k dosažení cíle byl používán pouze kvalifikovaný odhad. Za druhé

Tabulka 1.1: Typy bezpilotních prostředků

Typ	Anglická zkratka	Příklad
podvodní	UUV	MK 18 MOD 2 Swordfish
hladinové	USV	Elbit Systems Silver Marlin
pozemní	UGV	Google Self-Driving Car
vzdušný	UAV	General Atomics MQ-1 Predator
vesmírný		JPL Voyager 1 a 2

světové války byla vyvinuta jako první použitelná řízená střela s plochou dráhou letu Fiesler FI 103 V1. Navigace probíhala podobně jako i Ketteringova torpéda pomocí gyro-kompasu. Střela byla poháněna pulzačním motorem, a proto se pro výpočet vzdálenosti k cíli používaly otáčky malé vrtulky na přídi. Během studené války docházelo k rozvoji UAV především jako řízených střel. Začaly se však objevovat i výzvědné prostředky jako Lockheed D-21 [7]. V poslední době se pak dostávají do civilního prostředí jako systémy pro sledování, dopravu zboží, mapování, čistě pro zábavu a do budoucna je počítáno i s atmosférickými satelity.

V případě pozemních prostředků se historie začíná psát okolo roku 1920, kdy byly provedeny první zkoušky s auty na dálkové ovládání. Za druhé světové války se například používala německá pojízdná mina Goliath Sd.kfz 302. Ta byla řízena po drátu vojákem v zákopu pod nepřátelský tank, kde byla odpálena. Během padesátých a šedesátých let se objevily koncepty pro řízení aut pomocí speciálních vodících drátů umístěných ve vozovce, jež vyzařují signál. Plně automatická auta se začala objevovat během 80.let jako například Navlab z univerzity Carnegie Mellon. V současné době existuje již několik projektů od významných firem jako Google, Tesla nebo Honda, které slibují plně automatická auta pro běžný provoz již v blízké budoucnosti[6].

Všechny vesmírné prostředky snad kromě amerického Space Shuttle, jenž k přistání vyžadoval pilota, jsou koncipovány jako bezpilotní. Řízení tímto způsobem je nutné, protože většina z nich nemá lidskou posádku a u těch, jež ji mají je pilotování natolik náročné a rizikové, že se lidem s jejich pomalými reakcemi a nevypočitatelným chováním nemohou přenechat. Na tyto prostředky jsou kladeny speciální nároky, protože ve vesmíru je vakuum, obrovské rozdíly teplot, poletují v něm objekty s velice vysokou rychlostí a jsou vystaveny ionizující radiaci. Pro ochranu elektroniky před radiací se používá speciální radiačně tvrzená elektronika a pro kritické systémy se využívá několika identických počítačů, které většinově hlasují o výsledku. Komunikace je také výzvou, protože i při šíření rychlostí světla může trvat let signálu i několik hodin, což znemožňuje efektivní ruční dálkové ovládání nebo zásahy do řízení při kritických událostech.

Zatímco hladinové prostředky se nijak výrazně neprosadily, podvodních je veliké množství. Za první podvodní bezpilotní prostředky lze považovat ponorková torpéda, jejichž prvním zástupcem je torpédo Whitehead [27]. Torpéda ze začátku nebyla nijak naváděna a v podstatě po vypuštění jen v přímém směru plula ke svému cíli pomocí lodního šroubu poháněného stlačeným plynem. V dnešní době se běžně používají podvodní roboti k výzkumu podmořského života nebo k opravám hlubokomořských kabelů.

1.1.1 Komunikace

Komunikace se nejčastěji realizuje pomocí rádiového vysílání kvůli relativně nízké ceně a dostatečné flexibilitě. Rádiové vysílání má dobrý prostup vzduchem, dostatečnou datovou kapacitu a snadnou integraci. Nejlepší kvality přenosu bezdrátových informací se dosáhne, pokud mají přijímač a vysílač vzájemnou přímou viditelnost, bez překážek nebo rušení. Pokud je přímé spojení špatné nebo nemožné, tak lze využít sítě typu mesh, jež informace předávají přes retranslační stanice od vysílače k přijímači. Jako retranslační stanice mohou sloužit další prostředky, pozemní stanice nebo satelity. Dalším z problémů rádiové komunikace je náchylnost k rušení zdroji přírodními (počasí, radiace) nebo umělými (systémy pro elektronický boj). Útlum signálu pak znemožňuje komunikaci pod vodou. Běžně používané frekvence proniknou vodou řádově na jednotky centimetrů až metrů. Výjimkou je vysílání v pásmu ELF (3 Hz až 3 kHz), které dokáže proniknout stovkami metrů slané vody, ale jeho vysílače jsou neprakticky veliké a drahé[35]. Alternativou pro čistě podvodní komunikaci mohou být také systémy používající ke komunikaci ultrazvuk šířený vodou. Ke komunikaci pod vodou, pokud zařízení není plně autonomní, se používá nejčastěji kabelový svazek vedoucí z hladinového plavidla. Skrze kabelový svazek se zařízení nejenom ovládá, ale je mu skrze něj dodávána i energie a v případě problémů za něj může být i vytaženo na hladinu. Šifrování se také stává důležitou součástí komunikačních protokolů, aby se zabránilo útočníkovi v převzetí kontroly nad strojem.

1.1.2 Ovládání

Nejjednodušším systémem pro ovládání bezpilotních prostředků je ruční jednosměrné dálkové ovládání s přímou viditelností prostředku. U složitějších systémů se přidává kromě ručního ovládání i systém pro plánování a úkolování prostředků. K zadávání těchto povelů je již třeba obousměrná komunikace, počítač a případně speciální ovládací stanoviště podobně jako na obrázku 1.1. Prostředkem vysílané informace (telemetrie) mohou obsahovat výšku, rychlost, polohu, náklony, zbývající energii, data k probíhající misi a další. Pro lepší povědomí o situaci mohou tyto údaje být doplněny o video přenos nebo radarový odraz situace před vozidlem. Dálkové ovládání je velice složité, protože dochází k latencím v přenosových systémech a u větších vzdáleností hraje svou roli i samotná rychlost šíření elektromagnetických vln. Při dálkovém řízení také operátor přichází o rozhled a zpětnou vazbu v podobě setrvačných sil, vibrací a jiných jevů doprovázející řízení v kokpitu stroje. Komunikace je také náchylná k přerušení, a proto musí

Obrázek 1.1: Ovládací stanoviště dronu Predator MQ-1[18]

mít prostředky záložní procedury pro případ výpadku, aby nedošlo k jejich havárii.

1.1.3 Navigace a navádění

Nejstarší prostředky byly ručně ovládané operátorem, ale s jejich zvyšujícím se dosahem přišla na řadu potřeba jejich navigace a případně navádění na cíl. Nejstarší a stále používanými prostředky pro navigaci byly IMU (inertial measurement unit) jednotky, které se skládaly z gyroskopu, akcelerometru a někdy i magnetometru. Tímto systémem lze získat aktuální úhly natočení, směr a akceleraci. Po přidání počítače a integrováním dat lze vytvořit inerciální navigační systém, který dokáže sdělit polohu prostředku vzhledem k bodu nula, kterým je většinou startovací pozice. Tento systém má výhodu v tom, že je nezávislý na vnějších okolnostech, avšak jeho přesnost se se vzrůstající vzdáleností zhoršuje. Systémem používaným od pradávna byla astronavigace, kterou se během 60.let podařilo automatizovat a je používána dodnes u vesmírných prostředků a prostředků s vysokým stupně odolnosti proti rušení jako například u mezikontinentálních balistických střel. Díky citlivým kamerám je použitelná i za dne, ale v atmosféře ji omezuje počasí. Dalším používaným systémem navigace byl, zejména u střel s plochou dráhou letu, systém TERCOM (Terrain Countour Matching) [4].TERCOM pracuje na principu porovnávání výškových map uložených v paměti počítače s povrchem snímaným radiolokačním výškoměrem. Tento systém však vyžaduje nasnímání povrchu před letem a nelze jej použít v oblastech s malou členitostí terénu.

Dnes většinu navigačních systémů nahradila navigace pomocí družic (GNSS) jako je třeba GPS. Tyto systémy pracují na principu měření doby letu signálu z družic umístěných na MEO orbitě(2000 km - 35 786 km) k přijímači umístěném na prostředku. Pomocí trilaterace ze čtyř a více družic se pak dá zjistit přesná poloha a nadmořská výška s přesností na metry. Vzhledem k použité technologii lze tyto systémy použít pouze s přímou viditelností oblohy a jejich přesnost se při špatném počasí nebo stínění budovami snižuje. Systém GPS je pak pro civilní užití omezen na výšku 18 km a rychlost 514 $\mathrm{m\,s^{-1}}$ a také může být kdykoliv vypnut vládou USA [37].

1.1.4 Pohon a energie

V počátcích bezpilotních prostředků se k pohonu používaly většinou spalovací motor nebo stlačený plyn, což bylo možné, protože jejich navigační systémy byly čistě mechanické. S nástupem elektroniky však bylo třeba přidat zdroj napájení pro elektroniku a v případě potřeby hlavní pohonný systém. Pohonný systém je volen na základě prostředí, ve kterém se prostředek pohybuje a požadavkům na výdrž bez doplnění paliva. Pro zařízení, která běží čistě na elektřinu se k napájení používají baterie. Baterie mají nevýhodu v malé hustotě energie na hmotnost, proto se vyvíjejí alternativní možnosti jako například palivové články. K dobíjení se kromě elektrické sítě dají použít i alternativní zdroje jako jsou solární panely nebo v případě vesmírných sond radioizotopové reaktory. Nabíjení si pak dnešní prostředky mohou obstarat samy pomocí nabíjecího doku nebo bezdrátové nabíjecí podložky, ke kterým jsou schopny se samy navést a připojit. U větších prostředků je pak možno nasadit pohon se spalovacím, raketovým, proudovým motorem spolu s elektrickým generátorem pro napájení elektroniky.

1.1.5 Legislativa

Jako u každé technologie, která se úspěšně rozšiřuje, se i bezpilotní a zejména letecké prostředky dostaly do hledáčku vlád. Bezpilotní prostředky lze i zneužít k páchání nelegální činnosti jako je přeprava drog, doprava výbušnin, zasahování do letového prostoru nebo rušení bezdrátové komunikace. V České republice se provoz vzdušných prostředků řídí dle zákona č.49/1997Sb. o civilním letectví [21] a bezdrátová komunikace se řídí dle všeobecného oprávnění Českého telekomunikačního úřadu č.VO-R/10.05.2014-3.

Pozemní vozidla mají volnější regulaci, která jim umožňuje jezdit prakticky kdekoliv na soukromých pozemcích. Autonomní vozidla pro provoz na veřejných komunikacích jsou stále ve vývoji, ale pro jejich otestování je

třeba jízda v běžném provozu. Automobilky již dnes montují do aut pomocné prvky jako jsou adaptivní tempomaty nebo asistenty udržení se v jízdním pruhu, ale odpovědnost za samotné řízení je stále plně ponechána na řidiči. Plně automatická vozidla se pak musí řídit novými zákony, které ve většině světa ještě nebyly implementovány. V současné době se stále řeší filosofické otázky, kdo je odpovědný za nehody, a jak se má prostředek chovat při haváriích, kdy může svými zásahy ovlivnit, kdo zemře a kdo bude žít. V době psaní této práce umožňuje provoz autonomních vozidel v testovacím režimu jen hrstka států a žádný v běžném režimu[23].

1.2 Týmy robotů

Kooperace několika robotů (bezpilotních prostředků) je velice výhodná i za cenu nižší energetické efektivnosti a větší náročnosti jejich koordinace. Více robotů poskytuje redundanci při výpadku některého ze strojů. Stroje v týmu také mohou být specializovány pro řešení jednotlivých podúkolů, což jim umožňuje být jednodušší a tím i levnější. Větší počet malých robotů je také lépe manévrovatelný než jeden velký[36]. Na popularitě získávají takzvané roje robotů inspirované hmyzem žijícím v koloniích. Tyto roje obsahují identické stroje, které spolupracují na řešení zadaného úkolu. K jejich řízení se nepoužívá centrální systém, ale každý robot se řídí sám a udržuje svojí pozici vzhledem k ostatním okolo něj pomocí senzorů [16].

1.3 Augmentovaná realita

Augmentovaná realita je živý obraz reálného světa doplněný o různá senzorická data nebo počítačem generované objekty. Pravděpodobně nejstaršími příklady augmentované reality jsou reflexní zaměřovače (kolimátory) ze začátku století[30]. Téma augmentované reality je velice rozsáhlé, a proto se tato práce zaměřuje pouze na uživatelem nositelné přístroje takzvané Head Mounted Display (HMD).

Za první moderní systém augmentované reality je považován The Sword of Damocles virtual reality system z roku 1965 [34]. Systém obsahoval na teleskopickém rameni zavěšenou helmu obsahující binokulární poloprůhledné obrazovky, jež si uživatel nasazoval. Rameno kromě zavěšení helmy také snímalo pozici uživatele a náklon helmy, na jejímž základě počítač generoval obraz, který uživateli připadal jako staticky umístěný v prostoru i při pohybu uživatele. Uživatel tedy měl pocit, že počítačem generované objekty

jsou součástí reálného světa. Z tohoto principu pak vycházejí všechny HMD pro zobrazování virtuální i augmentované reality.

1.3.1 Zobrazování

HMD zařízení promítají obraz na obrazovku umístěnou pár centimetrů od očí a v podstatě zakrývají celý výhled. Kvůli blízkosti zobrazovačů k očím je zapotřebí vysokého rozlišení obrazovek, aby se zabránilo pixelaci obrazu. Virtuální realitu lze vytvořit dvěma způsoby, buď využijeme poloprůhlednou obrazovku nebo použijeme neprůhlednou. V případě neprůhledné obrazovky je reálný svět snímán pomocí kamer umístěných na HMD zhruba v pozici očí a jejich obraz je vykreslován jako nejspodnější vrstva vykreslované scény. V případě poloprůhledných obrazovek jsou používány technologie promítající obraz na poloprůhledné sklíčko nebo poloprůhledné obrazovky typu LCD nebo OLED. Poloprůhledné obrazovky mají výhodu v menším množství potřebného hardware, ale technologie poloprůhledných obrazů má stále problémy s dostatečným kontrastem a barevnou hloubkou. V současnosti jsou ve vývoji i kontaktní čočky obsahující display.

1.3.2 Sledování

Zobrazované augmentované objekty mohou být nezávislé nebo závislé na pozici pohledu uživatele. Nezávislé objekty většinou zobrazují jednoduché stavové informace. Pro zobrazování závislých objektů, které většinou nějak interagují s prostředím je třeba znát natočení a polohu HMD. Sledování rotace hlavy se realizuje pomocí IMU jednotky s magnetometrem, jež poskytuje náklon obrazovky a směr vůči magnetickému poli Země. Sledování pozice HMD a tím i jeho uživatele v prostoru je pak mnohem složitější a existuje k nim několik přístupů. Implementačně nejjednodušší a použitelné pouze v otevřeném prostranství je použití satelitní navigace, která má ale přesnost v řádu metrů.

Přesnější metody používají optické sledování kamerami. Nevýhodou tohoto řešení je veliká výpočetní náročnost zpracovaní obrazu a z toho vycházející nízká obnovovací frekvence nebo potřeba vysoce výkonného hardware. Polohu lze získat z analýzy kamer na HMD [20], jež snímají okolní prostředí nebo externí kamerou sledující referenční body typicky infračervené diody umístěné na HMD. Další systémy pak fungují na principu triangulace, kdy HMD vysílá elektromagnetické nebo akustické signály, které jsou zachycovány senzory se známou pozicí a z doby letu signálu se vypočítává pozice uživatele. Velice zajímavou sledovací technologií je Valve Lightnouse [28]. Tato technologie využívá dvou rotujících 60 Hz infračervených vysílačů vysí-

lajících ve dvou rovinách paprsek ve tvaru čáry. Na HMD jsou pak umístěny fotosenzitivní senzory. Ve chvíli, kdy jsou senzory excitovány, se tak získá úhel náklonu vysílače a data z IMU jednotky, z čehož se vypočítá pozice v prostoru.

1.3.3 Ovládání

Ovládání HMD není stále uspokojivě vyřešeným problémem. Jednoduché systémy využívají herní ovladače jako například gamepad, protože jsou uzpůsobeny tak, aby nepotřebovaly oční kontakt a uživatelé se jejich ovládání naučí velice rychle. Vyspělejší ovladače jsou vybaveny IMU jednotkou a případně i systémy pro sledování polohy. Oblíbeným ovládacím prvkem pro ovládání prostředí u systémů speciálně navržených pro augmetovanou realitu jsou dotykové plochy. Například u Epson Moverio je jediným ovládacím prvkem externí ovladač do ruky s klasickým touchpadem. Ovládání bez periferií je možné pomocí gest rukou nebo hlasem. K detekci gest se používá kamerový systém nebo odraz ultrazvuku jako u Leap Motion [15]. Ovládání gesty se zatím ukazuje jako problematické, protože gestikulace každého člověka je odlišná a jednotlivá gesta lze špatně mezi sebou rozlišit. Pro skutečně intuitivní ovládání se začíná používat hlasové ovládání v přirozeném jazyce. Stroje však stále nedosáhly na plné porozumění mluvené řeči, a tak se hlasové ovládání omezuje na jednoduché věty nebo předdefinované příkazy [19].

Kapitola 2

Výběr komponent sytému

Z přehledu dostupných technologií vyplynulo, že systém bude rozdělen mezi operátora a bezpilotní prostředky. Pro obě části bude vybrán potřebný hardware a následně pro ně bude napsán i speciální software.

2.1 Hardware pro operátora a prostředky

Obousměrná komunikace mezi operátorem a prostředkem je zajišťována bezdrátově pomocí radiových vln. Zatímco telemetrická data mají minimální datovou velikost, u video přenosu je naopak obrovská. Běžně tento problém modeláři řeší používáním dálkových ovládání pracujících v bezlicenčním pásmech a odděleným analogovým video přenosem na 5.8 GHz. Cena hardware pro analogový přenos videa je relativně vysoká a její propojení s augmentovanou realitou by bylo problematické, protože by vyžadovalo analogově-digitální video převodník k propojení s brýlemi pro augmentovanou realitu. Datové modemy na nízkých bezlicenčních frekvencích pak nemají dostatečnou datovou propustnost, proto bylo zvoleno Wi-Fi. Integrace Wi-Fi je snadná díky podpoře operačními systémy a cena hardware je nízká. Datová propustnost je také dostatečná i pro video přenos. Její nevýhodou je malý dosah a veliké rušení v zástavbě způsobené domácími routery.

Nejdůležitější hardwarovou komponentou používanou operátorem jsou brýle pro augmentovanou realitu. Cena dedikovaných zařízení je stále velice vysoká [9], proto byla vybrána dostupná alternativa. Touto alternativou se stal ColorCross (klon Google Cardboard) původně určený pro zobrazování virtuální reality. Pro přeměnu tohoto zařízení na zobrazovač augmentované reality stačí přidat živý obraz světa snímaný kamerou na zadní straně vloženého telefonu. Do ColorCross lze vložit prakticky jakýkoliv telefon, ale pro výslednou aplikaci je třeba vysoký grafický výkon, kvalitní IMU jednotka,

Obrázek 2.1: Brýle pro virtuální realitu ColorCross

obrazovka s vysokým rozlišením, GPS a Wi-Fi. Pro běh aplikace byl zvolen referenční telefon Google Nexus 5. Ovládání systému musí být prováděno ovládacími zařízeními, která nevyžadují velkou koncentraci, tak aby se operátor mohl plně soustředit na ovládání prostředku. K ovládání virtuální i augmentované reality existují různá dedikovaná zařízení, která jsou svázána s konkrétními brýlemi a mají vysokou cenu. Jako hlavní ovládací prvek byl zvolen generický gamepad, který je běžně používán konzolovými hráči a podobá se modelářským dálkovým ovladačům.

K bezpilotním prostředku je třeba mít připojený počítač, který zprostředkovává přenos dat, správu videa a ovládá prostředek. Za tímto účelem byl zvolen Raspberry Pi B [26], který splňuje všechny požadavky, je lehký a má malou spotřebu energie, což je zvláště přínosné u rotorových bezpilotních prostředků s nízkou nosností. Pro řízení samotných motorů a serv bezpilotního prostředku je použita platforma ArduPilot [5], připojená k počítači. K počítači je dále připojena digitální web kamera a USB Wi-Fi adaptér.

2.2 Operační systémy pro běh systému

Na trhu mobilních telefonů je v dnešní době zastoupena pouze hrstka operačních systémů. Prvenství v současné době patří Google Android následovaným Apple iOS a pak již jen systémy s minoritním podílem jako Ubuntu Phone, Firefox OS, Sailfish OS a Windows Phone [1]. Hned ze začátku

bylo jasné, že aplikace bude vysoce výkonově náročná a pro kvalitní zobrazení bude potřeba display s vysokým rozlišením. Dnešní výkonné telefony stále nevyrovnají výkonem stolním počítačů, proto musel být pro běh aplikace použit systém, který umožňuje běh nativních aplikací. Tento požadavek vyřazuje Firefox OS, protože se pro něj aplikace píší pouze pomocí JavaScriptu a HTML 5. Jako systém pro vývoj aplikace byl použit GNU/Linux, pro který ani iOS ani Windows Mobile neposkytují SDK. V době zahájení prací bohužel nebyl pro Ubuntu Phone ani Sailfish OS dostupný dostatečně výkonný hardware. Z dostupných operačních systémů byl tedy zvolen Google Android, který kromě vývoje v jazyku Java s využitím Android SDK podporuje i psaní nativních aplikací v C/C++ s využitím NDK a také je pro něj na trhu dostatek výkonných zařízení s obrazovkami s vysokým rozlišením. Raspberry Pi umístěné na prostředku má oficiální podporu pouze pro GNU/Linux, a proto byla zvolena distribuce Raspbian ve verzi Jessie.

2.3 Použité knihovny a software

Omezujícím parametrem pro výběr programovacího jazyku v aplikaci pro operátora byly zvolené systémy Android a GNU/Linux a také relativně nevýkonný hardware pro běh. Pro programování aplikace bylo zvoleno C++ protože je nativní, má menší spotřebu paměti než Java. K vytvoření aplikace jsou třeba knihovny pro hardwarově akcelerované kreslení grafiky, grafické uživatelské rozhraní, síťové spojení, kódování/dekódování videa a správy vstupů od uživatele.

Jako knihovna pro vykreslování grafiky byla použita OpenGL ES, protože je jako jediná zároveň podporována Androidem a GNU/Linuxem. Android obsahuje knihovny pro tvorbu uživatelského rozhraní, které jsou dostupné pouze skrze Java API a jsou systémově závislé. K napsání aplikace byl zvolen multiplatformní Qt Framework[24] ve verzi 5.5 a zejména jeho součást QtQuick sloužící pro psaní grafického rozhraní. QtQuick je použitelný s C++, ale také s JavaScriptu podobným jazykem QML. QML umožňuje rychlé skriptování rozhraní a jednoduché prototypování grafických komponent, ale jeho hlavní části jsou napsány v C++, aby dopad na výkon byl minimální. Rychlé prototypování je při vývoji této aplikace důležité, protože standardní, například desktopové ovládací prvky, jsou v augmentovaném uživatelském prostředí nepoužitelné a je třeba vytvořit nové.

Vzhledem k použití Wi-Fi bylo na přenosové síťové vrstvě na výběr pouze mezi protokoly TCP nebo UDP. Z těchto protokolů byt zvolen UDP, protože většina přenášených dat bude mít pouze krátkou platnost, jejich

Tabulka 2.1: Seznam poskytovatelů trojrozměrných map a jejich vlastnosti

Název	3D API	Pokrytí ČR	Web
Google Maps	NE	ANO	`www.maps.google.com`
Bing Maps	NE	NE	`www.bing.com/maps`
Nokia HERE	(placené)	ANO	`www.here.com`
OpenStreetMap	ANO	ANO	`www.osmbuildings.org`
Mapy.cz	NE	(vybraná města)	`www.mapy.cz`

nedoručení nebude představovat větší problém a bude třeba je velice často obnovovat. TCP se svou zárukou přenosu dat by jen zbytečně zatěžovalo přenosový kanál. Pro aplikační vrstvu je použit protokol MAVLink [2], který je přímo vyvinut pro komunikaci s bezpilotními prostředky a je nativně podporován autopilotem ArduPilot.

2.4 Mapy pro navigaci

Programy umožňující plánování tras pro bezpilotní prostředky obsahují mapové podklady pro jednodušší zadávání geografických souřadnic. Tyto mapy jsou většinou topografické nebo satelitní. Použití těchto map je již otestováno, a proto bylo dáno za cíl v tomto systému otestovat použití trojrozměrných map, které by měly přispět k lepší orientaci a také snazšímu plánování tras díky informacím o přesné výšce terénu. Trojrozměrné mapy lze získat od několika poskytovatelů uvedených v tabulce 2.1. Jednotliví poskytovatelé se liší v kvalitě map, pokrytí světa a také v dostupnosti API. Z výsledků vycházejí jako ideální OpenStreetMap, které ale nemají satelitní mapy a trojrozměrné mapy jsou většinou automaticky generované z geografických map a bez textur. Proto byly nakonec zvoleny Mapy.cz od Seznam.cz díky své liberální licenci, která je umožňuje použít za jakýmkoliv účelem [29].

Kapitola 3

Popis vytvořených programů

Systém je rozdělen na operátorskou aplikaci v telefonu jménem Looking Glass a aplikaci bezpilotního prostředku jménem Looking Glass UAS. Mezi aplikacemi je zajištěno síťové spojení s hvězdicovou architekturou, kde operátor je server a bezpilotní prostředky se k němu připojují jako klienti.

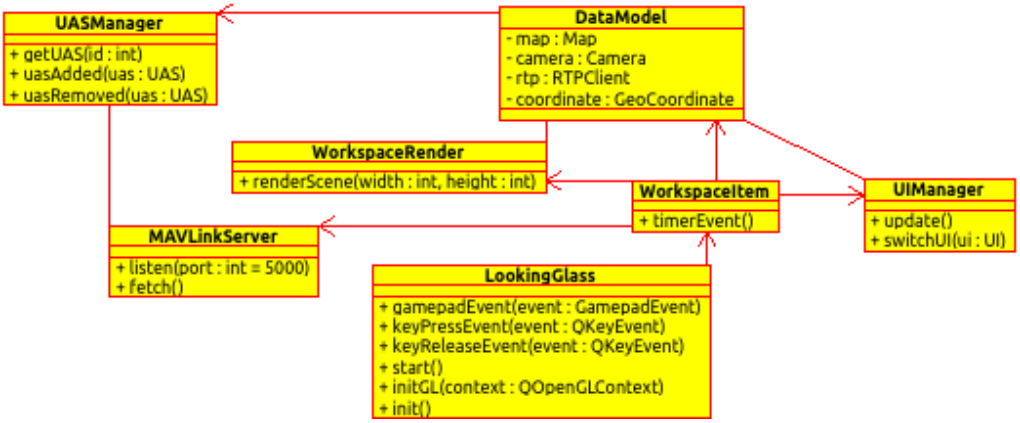
3.1 Aplikace Looking Glass

Součásti aplikace sloužící jako model a kontrolér jsou naprogramovány většinou v C++ s výjimkou přístupu k systémovým API Androidu, které jsou dostupné pouze v Java. K těmto částem se přistupuje přes JNI rozhraní, které propojuje JVM s nativním kódem. Grafické části aplikace byly vytvořeny výhradně v jazyce QML a jsou vykreslovány přes OpenGL. Výsledná aplikace se všemi závislostmi byla zabalena do Android instalačního balíčku .apk.

Aplikace operátora běží výhradně na mobilním telefonu. Kromě spuštění aplikace musí uživatel vždy zajistit nastavení gamepadu a síťového spojení. Zprovoznění spojení se provede přepnutím Wi-Fi telefonu do režimu hotspot, který slouží k připojení bezpilotních prostředků. Výchozí nastavení počítá s vytvořením sítě jménem LookingGlass se zabezpečením WPA2.

3.1.1 Architektura aplikace

Architektura aplikace, jejíž část je vyobrazená v diagramu 3.1, je velice složitá, a proto je zde prezentováno pouze její jádro. Základ architektury aplikace tvoří třída **LookingGlass**. Tato třída se stará o inicializaci programu a předávání událostí od uživatele do třídy **WorkspaceItem**. Inicializace programu byla rozdělena do několika kroků, protože program musí



Obrázek 3.1: Hlavní část architektury programu Looking Glass

čekat na inicializaci OpenGL frameworkem. V prvním kroku se v konstruktoru nastaví parametry pro QtQuick a OpenGL v hlavním vlákně. Dále se předá řízení frameworku a počká se na jeho zavolání metody initGL() z renderovacího vlákna informující o vytvoření OpenGL kontextu. Na závěr se asynchronně zavolá metoda init(), která provede finální inicializaci v hlavním vlákně a vytvoří sdílený OpenGL kontextu pro asynchronní nahrávání dat na grafickou kartu. Sdílené kontexty pro obě vlákna zamezují výpadkům snímků, ke kterým by docházelo pokud by muselo vykreslovací vlákno čekat na nahrání dat.

Třída **WorkspaceItem** je hlavním kontrolérem a **DataModel** hlavním datovým modelem celé aplikace. Kontrolér je priodicky každých 20 ms volán funkcí timerEvent(). Tyto konstantní updaty slouží k aktualizaci geografické pozice operátora, řídících obrazovek a synchronizaci packetů s třídou **MAVLinkServer**, která je blíže popsána v kapitole 4.5. Řídicí obrazovky slouží jako kontroléry obsluhující jednotlivé kontextové funkce programu. Třída **UIManager** má na starosti jejich správu, přepínání a přeposílání uživatelských událostí z **WorkspaceItem**. Bezpilotní prostředky, které naváží spojení se serverovou třídou **MAVLinkServer** jsou spravovány třídou **UASManager**. Ta má na starosti notifikaci zbytku programu o změnách v připojených prostředcích pomocí Qt signálů uasAdded() a uasRemoved(). Třída se také stará o jejich automatické odpojování v případě jejich neaktivity delší jak 5 s. Vykreslováním celé scény se zabývá třída **WorkspaceRenderer**. Ta provádí překreslení scény buď na povel z Qt frameworku (v případě, že se změnilo grafické rozhraní) nebo automaticky každých cca 17 ms pro dosažení konstantního framerate 60 Hz.

3.1.2 Paralelizace

Hlavním kriteriem při návrhu aplikace bylo dosažení maximálního počtu vykreslovaných snímků za sekundu a minimální latence spojení. Z tohoto důvodu je aplikace rozdělena do celkem čtyř vláken, kde každé obsluhuje svůj daný subsystém a neblokuje ostatní. Hlavní vlákno se stará o chod aplikace, vstup uživatele a přepínání mezi jednotlivými funkcemi aplikace. Grafické vlákno vytvořené Qt frameworkem se stará pouze o vykreslování scény na obrazovku telefonu. Hlavní vlákno se s vykreslovacím nijak nesynchronizuje kromě registrování nových grafických objektů, protože by to mohlo způsobovat trhání obrazu. Veškeré nastavení se děje atomickými změnami jednoduchých proměnných v datovém modelu. Ve vlastním vlákně běží také serverová část aplikace, která přijímá zprávy od klientů, které jsou každých 20 ms předávány přes kritickou sekci ke zpracování hlavnímu vláknu. Poslední vlákno se stará o dekódování videa přicházejícího od klienta a je spouštěno pouze v módu manuálního ovládání.

3.1.3 Pozemní řídicí stanoviště

Pro ovládání bezpilotních prostředků existuje několik specializovaných programů, takzvané Ground control station. Při vývoji systémů byly využívány pro nastavování aplikace QGroundControl a MissionPlanner, používané hobby modeláři k řízení a nastavování bezpilotních prostředků založených na protokolu MavLink. Tyto aplikace umožňují plánování misí, ruční ovládání prostředků a analýzu dat získaných při provozování prostředků. Jejich funkcionalita se stala vzorem při tvorbě této práce.

3.1.4 Uživatelské rozhraní

Uživatelské rozhraní bylo vytvořeno naprosto unikátně, protože technologie v oblasti zobrazování a ovládání augmentované reality jsou stále ve vývoji. Po vzoru head-up diplay je rozhraní silně kontrastní s převažující zelenou barvou, která je používána kvůli největší citlivosti lidského oka na tuto barvu [13]. Již při prvních testech rozhraní se ukázalo, že i přes relativně vysoké rozlišení obrazovky dochází výrazně k viditelnému rozpadání obrazu (pixelace) objektů. Tento problém výrazně omezuje minimální čitelnou velikost použitých fontů a tloušťky čar. Z tohoto důvodu jsou v rozhraní zobrazeny vždy pouze nezbytně nutné prvky a další funkcionalita je dostupná přes vyvolání kontextového menu. K ovládání programu je využita z velké části klávesnice, protože v Qt je implementována jako alternativa k ovládání ovládacích prvků místo myši. Události z gamepadu jsou pak jen

mapovány na ekvivalentní události klávesnice a injektovány do frameworku pomocí metody **QCoreApplication::sendEvent()**.

Nejbližším existujícím řešením k tomuto programu jsou pozemní řídící stanice viz. odstavec 3.1.3. Z nich bylo zjištěno, že aplikace musí obsahovat rozhraní pro plánování a manuální ovládání prostředků. K nim také musí být přidáno rozhraní pro monitorování skupiny bezpilotních prostředků a systém pro jejich výběr. Tato rozhraní byla rozdělena do tří samostatných obrazovek, mezi kterými operátor může přepínat.

3.1.4.1 Komponenta HUDMenu

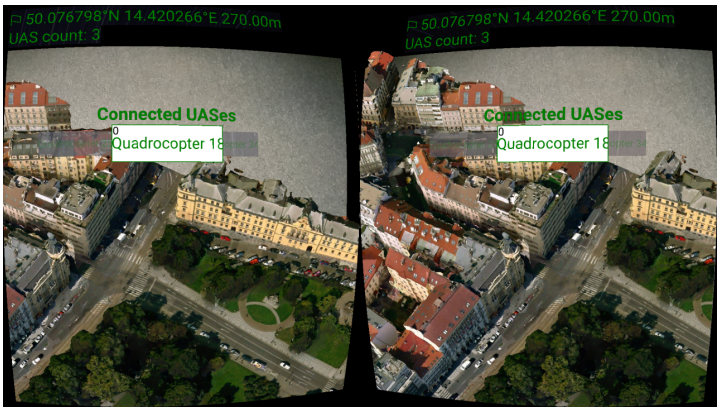
Kvůli velikostním omezením obrazovky bylo třeba vytvořit skryté kontextové menu. Proto byla vytvořena nová QML komponenta **HUDMenu** s položkami **HUDMenuItem**. Běžně používaný výpis položek by při větším množství byl nepoužitelný, proto byl vytvořen výpis umožňující zobrazení jen jeho části aniž by zhoršoval orientaci v něm. Menu je očíslovaná smyčka položek, kde poslední navazuje na první. Z menu je vždy viditelná jedna hlavní položka plus předchozí a následující položky v zákrytu za ní. V menu se lze pohybovat šipkami vlevo nebo vpravo tak, že se hlavní položka nahradí položkou předchozí nebo následující.

3.1.5 Obrazovky s uživatelským rozhraním

Základem všech obrazovek je C++ kontrolér, jenž je potomkem třídy **UI**, která je sama potomkem třídy **QQuickItem**. Takto vytvořený kontrolér je zaregistrován do QML jako QML komponenta, do které je vepsáno samotné grafické rozhraní. Tímto způsobem se dá striktně oddělit prezentace od implementace. Správu a předávání událostí od uživatele má na starosti třída **UIManager**.

3.1.5.1 Obrazovka UIOutlook

Outlook je hlavní obrazovka programu. Operátorovi umožňuje přímý výhled a zvýrazňuje pozice připojených bezpilotních prostředků. Prostředky jsou zvýrazněny bílou poloprůhlednou koulí, která se vykresluje operátorovi na souřadnicích vysílaných prostředkem. Kromě přímé viditelnosti může operátor využít trojrozměrnou mapu, popsanou v kapitole 4.1, s přehledem situace v okolí, která má zanesené pozice prostředků. Při přímém pohledu na prostředek jeho zvýrazňující koule zčervená. Tento stav slouží pro automatický výběr prostředku jedním stiskem. Prostředek lze také zvolit pomocí kontextového menu.



Obrázek 3.2: Obrazovka UIUASControl s trojrozměrnou mapou a kontextovím menu pro výběr bezpilotního prostředku

3.1.5.2 Obrazovka UIUAS

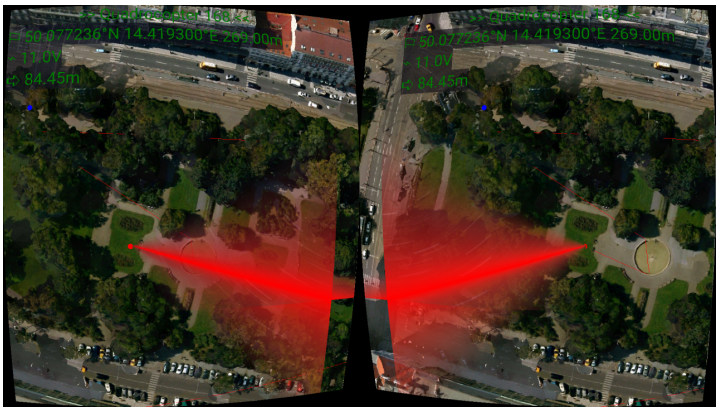
Tato obrazovka slouží k prohlížení telemetrie zvoleného prostředku a plánování jeho misí. Pomocí tlačítka nebo klávesy x lze vyvolat mapu, na které je zakreslena momentální naplánovaná mise. Pomocí kontextového menu lze tuto misi spustit nebo ukončit. Pomocí kontextového menu lze také vytvářet a mazat mise.

Mise se tvoří pomocí zdánlivě nehybné mapy a virtuálního laseru. Zadávání bodů do mapy je běžně děláno pomocí myši, a tak byl proveden pokus se simulací myši pomocí analogové páčky gamepadu. Tento způsob zadání fungoval, ale byl velice zdlouhavý. Lepším způsobem pro zadávání souřadnic se ukázalo zaměřování pomocí hlavy operátora. Operátor pohybem své hlavy zaměřuje virtuální laser na zvolené místo na mapě.

3.1.5.3 Obrazovka UIUASControl

UIUASControl slouží k manuálnímu řízení vybraného prostředku. K řízení prostředku může operátor využívat head-up display a živý obraz přenášený z prostředku, jenž nahrazuje augmentovaný obraz reálného prostředí. Ovládání prostředku se provádí pomocí gamepadu.

Účelem head-up display [31] je, u lidmi ovládaných prostředků, zobrazování důležitých telemetrických údajů v zorném poli uživatele. Díky tomuto zobrazení se může uživatel plně soustředit na řízení a neodpoutávat svůj zrak, aby se podíval na hodnoty přístrojů. Kromě telemetrie může být na display promítána i předpokládaná trajektorie nebo označení nepřátel. V aplikaci byly implementovány v QML čtyři znovupoužitelné komponenty.



Obrázek 3.3: Obrazovka UIUAS s laserem zadávajícím body mise

HUDAltitude
Umístěná na pravém okraji obrazu určuje nadmořskou výšku prostředku v metrech. Uprostřed je výrazně zobrazena aktuální výška a na vertikální nekonečné pásce odměřována výška, ke které se prostředek blíží.

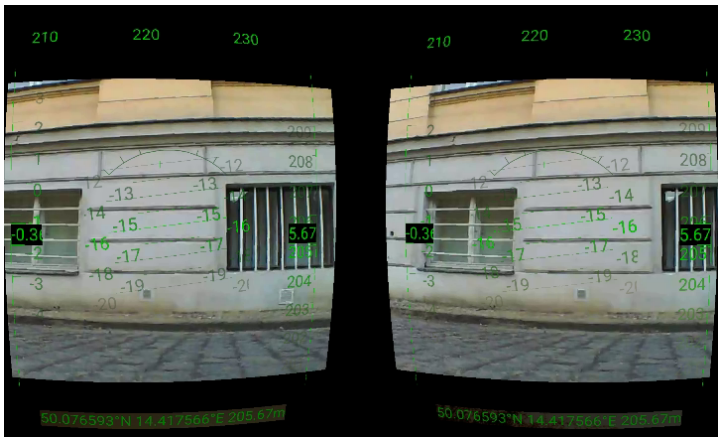
HUDSpeed
Se nachází na levém okraji obrazu a určuje rychlost prostředku v metrech za sekundu. Implementace je stejná jako u HUDAltitude.

HUDCompass
Na horním kraji obrazovky určuje natočení prostředku vůči magnetickému poli Země. Komponenta je zobrazována jako spojitá páska s 360 hodnotami odpovídajícími úhlu natočení.

HUDHorizon
Umělý horizont uprostřed zobrazuje operátorovi příčný a podélný náklon prostředku ve stupních. Podélný náklon je zobrazován jako páska s čárami reprezentujícími jednotlivé stupně náklonu. Příčný náklon je zobrazován jako náklon podélné pásky a ukazatele na polokruh zobrazující odklon od svislé osy.

3.2 Aplikace Looking Glass UAS

Bezpilotní prostředky jsou ovládány svojí vlastní aplikací vytvořenou pro běh na Raspberry Pi i normálním PC. Aplikace může fungovat v modu hardwarovém, kdy je k počítači připojen ArduPilot a také v módu virtuálním,

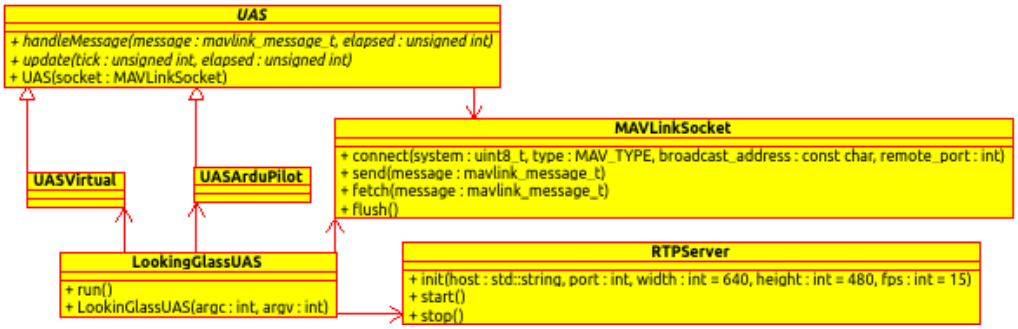


Obrázek 3.4: Obrazovka UIUASControl s head-up display

kde se simuluje chování bezpilotního prostředku. Zvolení módu aplikace se provádí pomocí přepínače při spuštění, kde `-r` spustí hardwarový mód a `-v` spustí virtuální. Ve virtuálním módu je implementována jednoduchá simulace kvadrokoptéry. Software Raspberry Pi byl upraven, aby automaticky spustil tuto aplikaci automaticky po startu systému, pomocí záznamu v `/etc/rc.local` B.3 a také se připojil k Wi-Fi síti vytvořené operátorem pomocí záznamu v souboru `/etc/wpa_supplicant/wpa_supplicant.conf` B.4.

3.2.1 Architektura aplikace

Hlavní třída **LookingGlassUAS** se stará o navázání komunikace s aplikací operátora a o virtuální zařízení kamery. Zavoláním metody metody connect() třídy **MAVLinkSocket** se otevře komunikační kanál. Systém začne vysílat packety do celé sítě a čekat na odpověď od serveru, po jejímž získání provede spojení. Po navázání spojení se třída **LookingGlassUAS** přepne do pracovního módu, ve kterém provádí konstantní aktualizace částí programu každých 10 ms. V těchto updatech zpracovává zprávy z **MAVLinkSocket**, které distribuuje kameře nebo jednomu z potomků třídy **UAS**. O přenos videa z kamery se stará třída **RTPServer**, která se spouští nebo vypíná na příkaz operátora. Třída **UAS** má potomka **UASVirtual** fungujícího jako plně virtuální bezpilotní prostředek a **UASArduPilot** fungující jako prostředník mezi ArduPilotem a programem. Komunikace s ArduPilotem probíhá fyzicky přes USB, ale ve skutečnosti probíhá přes virtuální sériový port.



Obrázek 3.5: Architektura aplikace Looking Glass UAS

Kapitola 4

Implementace důležitých součástí programů

4.1 Mapy

Mapy jsou velice užitečné při plánová misí, zjišťování pozice prostředku nebo jen udržení přehledu o situaci. Proto má operátor k dispozici mapu okolí synchronizovanou s jeho pozicí pomocí GPS. Na této mapě jsou také vyznačeny pozice jednotlivých bezpilotních prostředků.

4.1.1 Formát map

Pro Mapy.cz použité v tomto systému existuje pouze rozhraní implementované v JavaScript podporující pouze dvourozměrné vrstvy. Formát dat a způsob stahování map byl zjištěn analýzou webových stránek služby.

Z jejich studia zdrojových kódů vyplynulo, že mapy jsou rozděleny po dlaždicích na binární soubory s vektorovými daty a JPEG soubory s texturou. Jednotlivé dlaždice jsou na serverech adresovány[1] pomocí zoom, notrhing a easting. Zoom představuje detailnost dlaždice s hodnotou 1 až 18, vyšší čísla udávají vyšší rozlišení. Pro určování pozice na planetě Zemi se běžně používá formát zvaný WGS84, který určuje souřadnice na elipsoidu podobného tvaru Země. Alternativou k tomuto systému je UTM, který rozděluje planetu na síť šedesáti zón dle Mercatorova zobrazení [32]. Jednotlivé zóny v podstatě zobrazují části elipsoidu do roviny, což v rámci jedné zóny značně zlehčuje operace jako zjištění vzdálenosti dvou bodů. Právě tento systém je použit u Mapy.cz, kde northing a easting jsou x a y souřadnice v

[1] http://m1.mapserver.mapy.cz/3d/*zoom*-0*notrhing-easting*.(*jpg*/*bin*)

tomto systému v metrech. Reálná velikost dlaždice v metrech je dána rovnicí $2^{23-zoom}$. Pokud máme souřadnici v UTM, tak souřadnice dlaždice, ve které se bod nachází se vypočítá dle rovnic:

$$northing = \lfloor \frac{coord_{northong}}{tile\_size} \rfloor * tile\_size - 32$$

$$easting = \lfloor \frac{coord_{easting}}{time\_size} \rfloor * tile\_size - 32$$

Zatímco načítání textury dlaždice je jednoduché díky obrovskému množství knihoven, pro binární soubor bylo třeba napsat vlastní parser. Soubor začíná 56 B velikou hlavičkou obsahující identifikační ASCII řetězec souboru „BIN.MESH“, krajní hodnoty velikosti dlaždice, UV pozice textury a nakonec pořadí indexů, v jakém se dlaždice vykreslí. První dvě krajní hodnoty určují polovinu šířky a výšky dlaždice. Poslední hodnota pak určuje minimální a maximální nadmořskou výšku mezi body v dlaždici, ze které se dá vypočítat výška dlaždice. Následují souřadnice x_{in}, y_{in}, z_{in} jednotlivých vrcholů, které jsou uložené v rozsahu 0 až 65535. Získání reálných hodnot v metrech se provede výpočtem:

$$x = \frac{x_{in} * tile\_size}{65535}$$

$$y = \frac{y_{in} * tile\_size}{65535}$$

$$z = \frac{z_{in} * (tile\_height\_max - tile\_height\_min)}{65535}$$

Podobným způsobem jsou uloženy i souřadnice mapování textury. Pro OpenGL je však třeba je normalizovat do rozsahu 0.0-1.0. Trojrozměrné modely se skládají z trojúhelníků, které mohou mít společné vrcholy. Pro ušetření místa se tyto vrcholy neduplikují a místo toho se vytvoří seznam indexů, v jakém pořadí se má jaký bod vykreslit. K tomu také slouží poslední část souboru obsahující vykreslovací pořadí vrcholů.

Zdrojový kód 4.1: Pseudokód zobrazující strukturu souboru dlaždice mapy

```
 0B "BIN.MESH" (identification string)
 8B 3*double (min position xyz)
32B 3*double (max position xyz)
56B 1*uint16_t (number of vertices)
58B (number of vertices) * 3 * uint16_t (vertex x,y,z)
 ?B 1*uint16_t (number of texture coordinates)
 ?B (number of texture coordinates) * 2 * uint16_t
                (texture UV coordinates)
 ?B 1*uint16_t (number of vertex indices)
 ?B (number of vertex indices) * uint16_t
                (vertex index)
```

4.2 Kamera

Základem aplikací využívajících augmentovanou realitu je funkce zobrazující uživateli reálné prostředí. Obraz z reálného prostředí slouží jako podklad pro promítání dalších vrstev s informacemi. Dedikované přístroje pro augmentovanou realitu většinou využívají poloprůhledné hledí [9]. Na hledí jsou pak informace zobrazovány pomocí integrované poloprůhledné obrazovky nebo promítány malými projektory. V této práci je použita metoda, kdy se obraz reálného prostředí snímá pomocí přední kamery telefonu a následně se promítá jako nejspodnější vrstva augmentované reality na obrazovku.

Použitá metoda je oproti jednoduchému průzoru mnohem složitější a přináší několik softwarových a hardwarových problémů. Největším problémem je degradace obrazu oproti skutečnosti, která je spojená s hardwarovými vlastnostmi použitého přístroje. Kamery ani obrazovky stále nedosahují schopností lidského oka. Toto zhoršení lze pozorovat jako snížení rozlišení, snížení barevné hloubky a také zpomalení rychlosti ostření. Kamera umístěná na telefonu (jako u téměř všech dnes prodávaných modelů) také snímá pouze dvourozměrný obraz. Tyto vlastnosti nejdou v současné době s dostupnou techniku odstranit.

4.2.1 Implementace

V programu je vytvořeno abstraktní rozhraní kamery obalující čtyři různé implementace. Třída **Camera** poskytuje interface pro start kamery, notifikaci nových snímků a jejich kopírování do OpenGL textury. K nahrávání snímků do textury slouží metoda updateFrame(), která může být volána

pouze s aktivním OpenGL kontextem, protože v metodě probíhá překopírování obrazových dat z RAM na grafickou kartu. Toto rozhraní zapouzdřuje přístup k implementacím III a IV pracujících s nízkoúrovňovým, platformě závislým, API. Rozhraní kamery je na Androidu dostupné pouze prostřednictvím Java API. Pro obejití tohoto problému byl u těchto implementací vytvořen Java adaptér, který poskytuje implementacím přístup ke kameře skrze JNI. Kromě těchto implementací byly vytvořeny i I a II využívající standardní podporu knihoven. Cílem všech těchto implementací bylo nalezení systému s nejmenší latencí viz. odstavec 4.2.2.

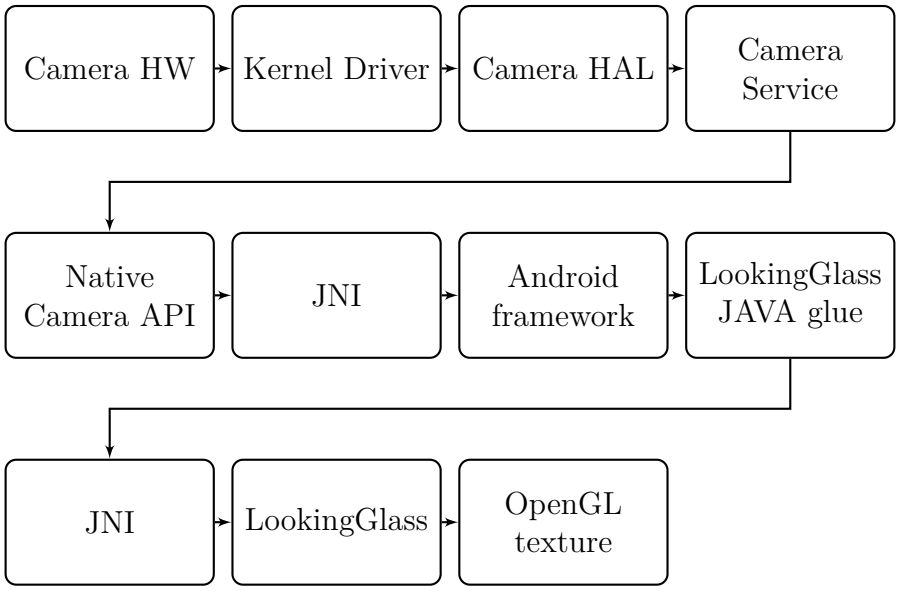
I **Android Camera standardní API**
Tato implementace je vytvořena pouze v Java. Přes Camera API je vytvořena instance kamera, jejíž výstup je zobrazován na instanci třídy **SurfaceView**. Tato třída je jako jediná vložená do hierarchie scény.

II **QtQuick Camera**
Qt framework na platformě Android umožňuje přístup ke kameře pouze skrze modul QtQuick. Do QtQuick scény je vložena QML komponenta typu Camera. Komponenta je zapnuta v režimu hledáčku a promítá obraz na QML komponentu VideoOutput.

III **Android camera s OES_EGL_image_external**
Android Camera API umožňuje nahrávání obrazu z kamery bez kopírování přímo do OpenGL textury pomocí rozšíření OES_EGL_image_external. Toto rozšíření umožňuje namapovat část paměti grafické karty do RAM podobně jako OpenGL pixel buffer object. Ke spuštění kamery je nutné v nativním kódu vytvořit OpenGL texturu, která se přes JNI předá do Java, kde slouží jako parametr do konstruktoru instance třídy **SurfaceTexture**. Objekt této třídy slouží jako virtuální cíl pro vykreslování obrazu z kamery, ale ve skutečnosti se nahrává do OpenGL textury v nativním kódu.

IV **Android camera s vlastním dekódováním**
Android bohužel neumožňuje zachytávání obrazových dat z kamery bez cílové zobrazovací plochy. V této implementaci je do scény vložena falešná instance třídy **SurfaceView** o velikosti 1x1px. Skutečný obraz se zachytává přes třídu **Camera.PreviewCallback**, která při každém novém snímku zavolá přes JNI nativní metodu a předá data snímku nativnímu kódu. Tato nativní metoda obrazová data zkopíruje a předá je do dedikovaného vlákna pro konverzi obrazu. Kamera dodává obraz ve formátu NV21, který se ve vláknu konvertuje na standardní RGB. Pře-

Obrázek 4.1: Softwarové vrstvy kamery



konvertovaný obraz ve formátu RGB se do OpenGL textury nahrává pomocí funkce glTexSubImage2D().

4.2.2 Latence zobrazování

Ve virtuální realitě je pro člověka přijatelná latence obrazu maximálně 20ms [14]. Již při první zkouškách programu se však ukázalo, že doba od zachycení snímku až po zobrazení je mnohem větší. Tato latence je způsobena hardwarem a také softwarem, který není optimalizován pro real-time snímání a zobrazování. Hardware obsahuje omezení v podobě propustnosti datových linek, rychlosti image signal procesoru, vyrovnávacích pamětí, ale i třeba rychlosti překreslení pixelů na obrazovce [10]. Softwarová latence je způsobena nutností projít přes několik vrstev aplikačního rozhraní popsaných v obrázku 4.1 a také tím, že všechny algoritmy aplikované na obraz mají asymptotickou složitost $O(width * height)$ nebo vyšší, vzhledem k velikosti obrazu.

4.2.2.1 Měření

Měření latence proběhlo na celkem čtyřech různých implementacích 4.2.1. K měření byla použita sestava sestávající se z vysokorychlostní kamery, mobilního telefonu a LED svítilny. Na telefonu vždy byla spuštěna jedna z implementací zobrazující obraz z přední kamery s rozlišením 1280x960px

Tabulka 4.1: Naměřená latence mezi událostí a jejím promítnutím na obrazovku.

Měření	Impl. 1	Impl. 2	Impl. 3	Impl. 4
1.	50	52	60	33
2.	44	57	24	63
3.	24	39	25	58
4.	56	35	26	32
5.	84	31	30	42
Průměr	51.6	42.8	33.0	45.6
Latence	$\approx$ 216 ms	$\approx$ 179 ms	$\approx$ 138 ms	$\approx$ 191 ms

přímo na obrazovku telefonu. Obraz na obrazovce byl snímán vysokorychlostní kamerou při 240 snímcích za sekundu, což odpovídá závěrce o délce cca 4.2 ms. Do záběru vysokorychlostní kamery i kamery v telefonu byla vložena LED svítilna, která byla několikrát po sobě rozsvícena a zhasnuta. Výsledný záznam z vysokorychlostní kamery byl po ukončení měření rozdělen na jednotlivé snímky. V těchto snímcích byly hledány posloupnosti, které začínaly rozsvícením LED svítilny a končily zobrazením světla na obrazovce mobilního telefonu. Výsledné délky posloupností, spolu s přibližnou latencí pro každou implementaci, jsou uvedeny v tabulce 4.1.

4.2.3 Vyhodnocení

Testy popsané v kapitole 4.1 ukázaly, že nejpomalejší způsob vykreslování vykazuje implementace používající čisté Android Java API. Implementace II a IV mají velice podobný čas, protože z analýzy kódu Qt vyplynulo, že pro získávání a vykreslení obrazu používají podobný postup. Nejrychlejším způsobem pro zobrazování obrazu kamery je implementace využívající OES_EGL_image_external. Toto vítězství je pravděpodobně způsobeno nízkoúrovňovou optimalizací dekódování obrazu a jeho nahrávání do OpenGL textury.

4.3 Gamepad

Qt framework obsahuje aplikační rozhraní pro přístup k periferiím jako myš, klávesnice nebo dotyková obrazovka. Mezi podporovaná rozhraní nepatří herní zařízení jako gamepad nebo joystick. Protože hlavním ovládacím zařízením pro ovládání programu je gamepad, bylo třeba vytvořit vlastní implementaci, sloužící pro přístup k této periferii.

Pro přístup ke gamepadu slouží třída **Gamepad**, která je potomkem jedné z implementací pro konkrétní platformu. Třída poskytuje metodu init(), jež slouží k nalezení a nastavení gamepadu v aplikaci a Qt signál gamepadEvent. Signál je spuštěn při každé události zařízení a jako parametr má třídu typu **GempadEvent**. Tato třída poskytuje informace kdy a jaký typ události nastal. Třída podporuje události typu BUTTON_PRESS / BUTTON_RELEASE, AXIS (pohyb analogových páček) a PEDAL (analogová tlačítka ovládaná ukazováky).

4.3.1 Linux

Implementace pro Linux je založena na manageru zařízení pro Linuxové jádro zvaném udev. Udev spravuje soubory zařízení ve složce `/dev` a informuje uživatelský prostor o jejich připojení nebo odpojení. Program naslouchá monitoru udev a čeká na událost od subsystému typu „input“. Při každé události se ověří, jestli je typu ID_INPUT_JOYSTICK, který značí přítomnost herního zařízení. Po tomto ověření se zařízení přidá nebo odebere z programu dle typu události. Udev vytvoří pro každý připojený gamepad soubor zařízení jménem `event` s číslem zařízení ve složce `/dev/input`. Po otevření tohoto souboru je možné číst události gamepadu pomocí standardních IO funkcí. Ze souboru jsou události čteny ve formátu struktur typu **struct** input_event. Pomocí maker v hlavičkovém souboru `linux/input.h` se z této struktury dekóduje konkrétní událost zařízení.

4.3.2 Android

Další z mnoha oblastí, kde se Android odlišuje od standardních linuxových distribucí je i správa zařízení. Android NDK obsahuje rozhraní pro přístup ke klávesnici, ale prozkoumáním hlavičkových souboru v NDK bylo zjištěno, že gamepad je podporován až od verze NDK 8b. Proto byla zvolena implementace v Java SDK a přes rozhraní JNI přenos událostí do C++. V Java třídě **LookingGlassActivity**, která je potomkem třídy **Activty**, jsou přetíženy metody dispatchGenericMotionEvent() a dispatchKeyEvent(). Metoda dispatchGenericMotionEvent() je systémem volána v případě ovládacích prvků, které mají uživatelsky nastavitelnou intenzitu. Tyto prvky jsou například analogové páčky, které vrací intenzitu náklonu oproti výchozí poloze. Metoda dispatchKeyEvent() slouží k zachytávání obyčejných tlačítek, která mění stav pouze mezi stisknuto/uvolněno.

4.4 Trojrozměrný obraz

Stereopsie je termín označující vidění s vnímáním hloubky neboli trojrozměrné vidění [17]. Tohoto efektu je dosaženo pomocí dvou nezávislých kamer(očí), které snímají scénu pod různými úhly. Trojrozměrné vidění umožňuje lepší orientaci v prostoru a jeho simulace je nezbytným předpokladem pro vytváření uvěřitelných virtuálních objektů.

K vytvoření simulace je použito brýlí pro virtuální realitu, které využívají mobilní telefon jak pro zobrazování obrazu, tak jako výpočetní jednotku. Brýle se skládají z držáku telefonu, pásků pro připevnění na hlavu a čoček. Přestože je display v brýlích velice blízko očím, pozorovací úhel je velice malý. Pro rozšíření pozorovacího úhlu jsou tedy použity bikonvexní čočky.

4.4.1 Parametry zařízení

Pro správné vykreslování obrazu je třeba znát několik parametrů brýlí pro virtuální realitu a také obrazovky, na které je obraz zobrazován. Parametry obrazovky se dají jednoduše zjistit zavoláním systémových funkcí v telefonu. Výrobci dodávaná dokumentace k použitým brýlím neobsahuje všechny potřebné parametry, proto byly k jejich získání použity alternativní metody a jejich výsledky byly zaneseny do tabulky 4.2.

Parametry pro Google Cardboard byly získány pomocí dekompilace Cardboard SDK [11] programem Java Decompiler [8]. Procházením výstupu bylo zjištěno, že rozměry Google Cardboard jsou uloženy v originálním zdrojovém kódu v souboru `CardboardDeviceParams.java` a radiální koeficienty zkreslení čoček v `Distortion.java`.

U brýlí ColorCross byla situace ztížená, protože čínský výrobce nezveřejnil vůbec žádné parametry. Proto byly všechny parametry kromě radiálních koeficientů změřeny přímo na zařízení pomocí posuvného měřítka. Pro zjištění radiálních koeficientů byl vytvořen program Calibration. Tento program využívá část kódu programu Looking Glass, aby zobrazil na obrazovce uniformní mřížku se stejným zkreslením. Tato mřížka pak při pohledu skrz brýle byla zakřivena působením barelové distorze čoček, jejíž radiální koeficienty je třeba najít. Radiální koeficienty jsou nalezeny, pokud je mřížka, na níž je hleděno skrze brýle a na níž je aplikována inverze distorze opět narovnána. V programu Calibration jsou tedy pomocí kláves, na k telefonu připojené klávesnici, měněny hodnoty koeficientů dokud tato situace nenastane.

Tabulka 4.2: Parametry brýlí pro virtuální realitu

Parametr	Google Cardboard	ColorCross
Osová vzdálenost očí	60 mm	65 mm
Střed čočky-spodek obrazovky	35 mm	35 mm
Průměr čoček	25 mm	34 mm
Čočka-obrazovka	37 mm	75 mm
Čočka-oko	11 mm	25 mm
Radiální koeficienty zkreslení K_0, K_1, K_2, K_3	1.0, 250, 50000, 0.0	1.0, 128, 256, 0.0

Tabulka 4.3: Parametry obrazovky Nexus 5

Parametr	Obrazové body	Fyzická velikost
Výška	1080 px	57 mm
Šířka	1920 px	102 mm

4.4.2 Renderování

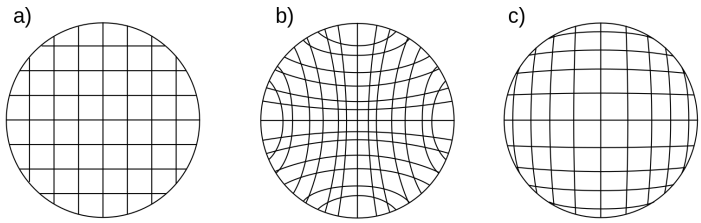
Obraz viditelný v brýlích se skládá ze tří vrstev, na něž je aplikován post-processing. Aplikace obsahuje OpenGL framebuffer o velikosti obrazovky, kde každá polovina slouží pro jedno oko. Jako první se vykresluje vrstva s kamerou. Druhá následuje vrstva s trojrozměrnými objekty a jako poslední se aplikuje uživatelské rozhraní.

4.4.2.1 Distorze obrazu

Čočky v brýlích způsobují takzvanou radiální barelovou distorzi obrazu. Tato distorze obrazu je dána obecnou rovnicí:

$$distort(r) = r * (K_0 + K_1 r^2 + K_2 r^4 + ...)$$

Parametr r je vzdálenost pixelu od středu čočky a K_n jsou radiální distorzní koeficienty. Výsledkem této rovnice je nová vzdálenost, která odpovídá pozici viděné skrze čočky. Cílem programu je distorzi neutralizovat, čehož se dosáhne pomocí poduškové inverzní distorze. Hodnota vzdálenosti pro inverzní distorzi je dána rovnicí $f(r) = r - r * (K_0 + K_1 r^2 + K_2 r^4 + ...)$. Pro výpočet této rovnice neexistuje analytické řešení, a proto bylo použito iterativní 4.2 pomocí metody sečen. Jako počáteční hodnoty byly zvoleny $poloměr/0.9$ a $poloměr * 0.9$, mezi kterými se nachází řešení rovnice.



Obrázek 4.2: Distorze obrazu způsobený čočkami a) rastr b) poduškovité zkreslení b) barelové zkreslení [22]

Zdrojový kód 4.2: Pseudokód pro výpočet inverzní distorze

```
float distortInverse(float radius)
{
  float r0 = radius / 0.9f;
  float r1 = radius * 0.9f;
  float dr0 = radius - distort(r0);
  while (abs(r1 - r0) > 0.0001f)
  {
    float dr1 = radius - distort(r1);
    float r2 = r1 - dr1 *
      ((r1 - r0) / (dr1 - dr0));
    r0 = r1;
    r1 = r2;
    dr0 = dr1;
  }
  return r1;
}
```

4.4.2.2 Geometrie obrazu

Z parametrů brýlí a telefonu je třeba vypočítat pozici a poduškovou distorzi obrazu tak, aby viditelný obraz zabíral co největší plochu, simuloval 3D a zároveň neobsahoval optické vady. Pro správné zobrazení obrazu je nejdříve nutné získat pozorovací úhly. Ideální pozorovací úhel bez čoček lze spočítat pomocí rovnice:

$$ideal\_fov\_angle = \arctan 2(lens\_diameter, eye\_to\_lens\_distance)$$

Dále je třeba vypočítat pro oči ideální pozici obrazu na obrazovce. Pro

levé oko se krajní pozice obrazu vypočítají dle rovnic:

$$eye\_to\_screen = eyeToLensDistance + screenToLensDistance \quad (4.1a)$$

$$edge_{left} = distort(\frac{screen_{width} - interpupillaryDistance}{2}) \quad (4.1b)$$

$$angle_{left} = \min(ideal\_fov\_angle, \text{atan2}(edge_{left}, eye\_to\_screen)) \quad (4.1c)$$

$$position_{left} = \tan(angle_{left}) * eye\_to\_screen \quad (4.1d)$$

$$edge_{right} = distort(\frac{interpupillaryDistance}{2}) \quad (4.1e)$$

$$angle_{right} = \min(ideal\_fov\_angle, \text{atan2}(edge_{right}, eye\_to\_screen)) \quad (4.1f)$$

$$position_{right} = \tan(angle_{right}) * eye\_to\_screen \quad (4.1g)$$

$$edge_{lower} = verticalDistanceToLensCenter \quad (4.1h)$$

$$angle_{lower} = \min(ideal\_fov\_angle, \text{atan2}(edge_{lower}, eye\_to\_screen)) \quad (4.1i)$$

$$position_{lower} = \tan(angle_{lower}) * eye\_to\_screen \quad (4.1j)$$

$$edge_{upper} = screen_{height} - verticalDistanceToLensCenter \quad (4.1k)$$

$$angle_{upper} = \min(ideal\_fov\_angle, \text{atan2}(edge_{upper}, eye\_to\_screen)) \quad (4.1l)$$

$$position_{upper} = \tan(angle_{upper}) * eye\_to\_screen \quad (4.1m)$$

(4.1n)

Z těchto vypočítaných pozic se spočítá fyzická velikost obrazu na obrazovce a bod nacházející se ve středu oka:

$$left\_eye_{width} = position_{left} + position_{right} \quad (4.2a)$$

$$left\_eye_{height} = position_{lower} + position_{upper} \quad (4.2b)$$

$$left\_eye_{center_x} = position_{left} \quad (4.2c)$$

$$left\_eye_{center_y} = position_{lower} \quad (4.2d)$$

4.4.2.3 Post-processing

Poslední procedurou před vykreslením scény na obrazovku je aplikace distorzní korekce na framebuffer obsahující celou scénu. K tomuto účelu je využit speciální model uniformní dvourozměrné mřížky, zvlášť pro každé oko, na něž se aplikuje framebuffer jako textura. Mřížka v každém bodě obsahuje

koordináty textury aplikované na tento objekt. Tyto koordináty je aplikována podušková distorze, jež neutralizuje zakřivení způsobené čočkami. Koordinát textury v bodě se vypočítá tak, že nejprve se stanoví koordináty textury v původním obraze:

$$texture_{width} = right\_eye_{width} + left\_eye_{width} \tag{4.3a}$$
$$texture_{height} = right\_eye_{height} + left\_eye_{height} \tag{4.3b}$$
$$uTexture = \frac{column}{(columns - 1)} * \frac{left\_eye_{width}}{texture_{width}} \tag{4.3c}$$
$$xTextureEye = uTexture * texture_{width} \tag{4.3d}$$
$$vTexture = \frac{row}{(rows - 1)} * \frac{left\_eye_{height}}{texture_{height}} \tag{4.3e}$$
$$yTextureEye = vTexture * textureHeightM \tag{4.3f}$$

Ze získaných fyzických pozic je třeba vypočítat jejich vzdálenost od středu čočky. Na tuto vzdálenost je aplikována funkce inverzní distorze:

$$rTexture = sqrt(xTextureEye^2 + yTextureEye^2) \tag{4.4a}$$
$$textureToScreen = \frac{distortInverse(rTexture)}{rTexture} \tag{4.4b}$$

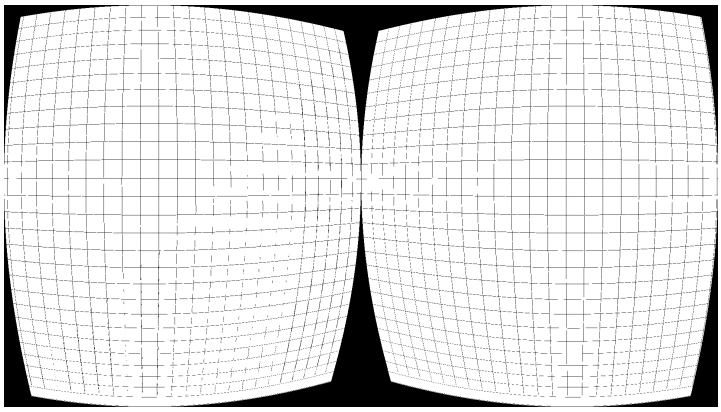
Výsledná hodnota se použije pro interpolaci pozice a následný převod do obrazových souřadnic. Nakonec se pak koordináta normalizuje do rozsahu $0.0 - 1.0$, jenž je podporován OpenGL u souřadnic textury.

$$uScreen = \frac{(xTextureEye * textureToScreen)}{screenWidthM} \tag{4.5a}$$
$$vScreen = \frac{(yTextureEye * textureToScreen)}{screenHeightM} \tag{4.5b}$$
$$u = (2.0 * uScreen - 1.0) \tag{4.5c}$$
$$v = (2.0 * vScreen - 1.0) \tag{4.5d}$$

Aplikací na všechny body se dosáhne obrazu poduškovitého zakřivení a tím i nezkresleného obrazu v brýlích.

4.5 Komunikace

Komunikační systém má standardní architekturu programů klient (Looking Glass UAS) - server (Looking Glass). Ke komunikaci mezi uzly se



Obrázek 4.3: Distorze obrazu aplikovaná na obraz pro vyrušení efektu čoček

využívá protokol MAVLink, který je v případě video přenosu doplněn protokolem RTP. Knihovna pro protokol MAVLink se v podstatě skládá pouze z hlavičkových souborů jednotlivých zpráv. Každá zpráva obsahuje unikátní identifikaci odesílajícího systému a typ komponenty, která zprávu odeslala. Implementační detaily, jako výběr přenosového kanálu, vytvoření spojení a posílání zpráv, jsou přenechány na programátorovi. K naprogramování síťového kódu obou programů byly využity pouze standardní BSD sockety.

Prvním krokem k úspěšné komunikaci je navázání spojení, pokud možno bez nutnosti nastavování. Klientská aplikace může běžet na počítači s několika síťovými rozhraními jako ethernet, localhost nebo Wi-Fi. Pro automatické spojení program otevírá socket všech dostupných rozhraní, na kterých vysílá zprávu typu mavlink_heartbeat_t všem počítačům v lokální síti. Toto vysílání je prováděno pomocí IP multicast. IP multicast umožňuje zasílání IP datagramů více počítačům v lokání síti bez znalosti jejich adresy. Na rozdíl od broadcast jsou datagramy zasílány pouze těm, kteří jsou zaregistrováni ve stejné skupině jako vysílač [12]. Server také otevírá sockety na všech rozhraních a je zaregistrován ve stejné multicast skupině jako klient. Server pak zaregistruje každého neznámého klienta. K udržení spojení je každou sekundu odesílána zpráva mavlink_heartbeat_t jak ze serveru tak z klienta. Pokud tyto zprávy nedojdou po dobu delší pěti sekund, tak je spojení ukončeno oběma konci.

Jak na klientovi tak na serveru, běží správa spojení v odděleném vlákně, aby případné náročnější úlohy nepřerušovaly doručování zpráv. K odesílání zpráv se používá vyrovnávací paměť, do které se ukládají zprávy k odeslání. K jejich samotnému odeslání dochází při dosažení MTU velikosti (obyčejně 1500 B) nebo na žádost hlavního vlákna. Tímto způsobem se předchází fragmentování datagramů a také šetří režijní náklady, které by si vyžádalo ode-

sílání každé zprávy zvlášť. Příchozí zprávy se ukládají do fronty v síťovém vlákně. Z této fronty jsou v kritické sekci zprávy předávány do hlavního vlákna ke zpracování.

4.6 Streamování videa

K ovládání bezpilotního prostředku uživatel potřebuje dostatek informací o jeho aktuální stavu. Pouhý přenos informací o směru nebo rychlosti není dostatečný, a proto má Looking Glass implementován real-time přenos obrazu z kamery umístěné na prostředku do brýlí. Přenos obrazu je velice datově náročný, což je dále zhoršeno využitím bezdrátového spojení. Pro minimalizaci objemu přenesených dat bylo třeba využít ztrátovou real-time video kompresi obrazu z kamery. Real-time komprese videa je výpočetně velice náročná, a proto bylo nutné se omezit pouze na hardwarově podporované formáty. Pro hardwarovou akceleraci Raspberry PI obsahuje VideoCore IV® Multimedia Co-Processor, který podporuje pouze kompresi formátu H.264[3]. Formát H.264 dosahuje dobrého kompresního poměru a je široce podporován. Alternativou by mohly být novější formáty jako H.265 nebo VP9, které mají ještě lepší kompresní poměr, ale menší podporu ze strany výrobců [33].

Na rozdíl od normálně přehrávaného videa má real-time přenos několik odlišností, díky kterým se dá snížit datový tok a latence. Jakýkoliv snímek kromě toho nejnovějšího je irelevantní. V zájmu rychlosti nemá smysl odesílateli potvrzovat doručení snímku. Příchozí snímky není třeba synchronizovat ani časovat, protože přicházejí v rozestupech, ve kterých je zachytila kamera.

Díky těmto zjednodušením bylo možno použít protokol UDP s protokolem pro přenos multimédií skrze IP sítě jménem RTP. Společně s RTP se často používají i protokoly RTSP a RTCP. RTSP se stará o navázání spojení a vyjednává video formát. Tento protokol použit nebyl, protože RTSP používá protokol TCP, který by zatěžoval linku a formát videa lze nastavit ručně v kódu programu před spojením. RTCP se používá ke zjišťování statistické kvality přenosu. Tento protokol také nebyl použit, protože by zatěžoval linku a také vzhledem k nemožnosti změny kvality videa za běhu by byl zbytečný. Implementace přenosu je rozdělena na klienta (telefon), který video dekoduje a zobrazí uživateli a server (Raspberry Pi), který vezme obraz z kamery a zakódovaný ho odešle.

4.6.1 Video Server

Video se odesílá rychlostí patnáct snímků za sekundu v rozlišením 640x480 pixelů, pouze na žádost operátora. K odesílání videa se používá knihovna OpenMAX pro hardwarové kódování obrazu a GStreamer, jenž se stará o celý proces odesílání videa. Ke spuštění GStreameru se používá konfigurační pipelina [2], která postupně vezme obraz z kamery, zakóduje ho pomocí knihovny OpenMAX do H.264 a pošle výsledek protokolem RTP klientu.

4.6.2 Video Klient

Na straně klienta je třeba video stream složit a převést komprimovaný obraz na OpenGL texturu, která je použita pro vykreslení na obrazovku telefonu. Při přepnutí na manuální ovládání program vyšle požadavek na zapnutí kamery na UAS a zároveň nastartuje v separátním vlákně přijímač obrazu. Přijímač přijímá video packety a skládá je do komprimovaných snímků. Složené snímky jsou předávány dekodéru, od kterého si vykreslovací vlákno žádá o výsledné obrázky.

Protože Android nepodporuje čisté RTP nativně, tak bylo rozhodnuto napsat vlastní minimální implementaci RTP protokolu, jejíž výstup je přímo předáván do dekodéru.

4.6.2.1 Přenos obrazu

Protokol RTP byl u klienta implementován bez knihoven, protože Android nepodporuje jeho čistou formu bez RTCP. Implementace byla provedena dle RFC 6184 [25], který definuje transportní formát pro video zakódované pomocí H.264.

Každý RTP packet obsahuje 12B hlavičku se sekvenčním číslem, časovým razítkem a dalšími informacemi. Ta je následována 1B fragment unit indikátorem, který obsahuje typ packetu odpovídající typům framů v H.264. Implementace podporuje typ identifikátoru 1 až 23, který obsahuje celý jeden frame a typ 28 obsahující fragmentovaný frame. U typu 28 se dále přečte 1B fragment unit hlavička, která rozliší jestli tento packet obsahuje začátek, prostředek nebo konec fragmentovaného framu, který je potřeba složit. Po přečtení těchto informací následují vždy surová data, která se skládají do výsledného framu a předávají dekodéru.

[2] v4l2src device=/dev/video0 ! video/x-raw,width=640, height=480, framerate=15/1 ! omxh264enc target-bitrate=300000 control-rate=variable ! rtph264pay config-interval=1 pt=96 ! udpsink host="host"port="port"sync=false

4.6.3 Dekódování

K dekódování videa na platformě Android bylo využito rozhraní MediaCodec. Toto rozhraní je v Java přítomno od první verze Android, ale jeho nativní verze až od verze 5.0. Dekódování přes toto rozhraní je možné napsat plně v nativním kódu, pokud je dostačujícím výstupem bitmapa. Pro vykreslení je, ale použita přímá konverze do OpenGL textury. Ruční konverze bitmapy na texturu by vyžadovala převod výstupu z dekodéru ve formátu YUV do RGB a následné nahrání do OpenGL, což by bylo časově náročné. MediaCodec také umožňuje ukládat výstup dekodéru přímo do OpenGL textury díky rozšíření GL_TEXTURE_EXTERNAL_OES. Ke zprovoznění této přímé konverze je třeba vytvořit OpenGL texturu v nativním kódu a s její pomocí vytvořit objekt typu **Surface** jako cíl dekodéru. K jeho vytvoření a monitorování změn obsahu textury byla naprogramována Java třída **X264Decoder**, protože třída **Surface** je dostupná pouze v SDK.

Na platformě GNU/Linux byla zvolena k dekódování knihovna libavcodec. Knihovna přijímá zakódované snímky a produkuje obraz ve formátu YUV420. Tento obraz je konvertován do RGB a následně nahrán do OpenGL textury, kde je využit při renderování obrazu.

Kapitola 5

Experimenty

5.1 Softwarová simulace

K otestování uživatelského rozhraní výsledného systému byla použita softwarová simulace. V této simulaci je reálný prostředek nahrazen virtuální kvadrukoptérou. Tato simulace je prováděna aplikací LookingGlassUAS se zapnutým virtuálním prostředkem, který běží na desktopovém počítači. Obraz z kamery prostředku byl nahrazen obrazem z webové kamery daného počítače. Při provádění testu měli uživatelé nasazené brýle ColorCross s telefon Google Nexus 5 a drželi gamepad Xiaomi Wireless Gamepad.

5.1.1 Testování

Ovládání tohoto systému je značně atypické od známých rutin pro ovládání počítačů nebo dotykových zařízení. Proto byl pro uživatele sestaven krátký návod k jeho ovládání a diagram s popisem funkcí gamepad 5.1.1.1 . Kromě toho jim byl také poskytnut krátký přehled jednotlivých částí systému a popis fungování augmentované reality.

5.1.1.1 Návod

Systém se skládá z brýlí obsahujících mobilní telefon. Aplikace v telefonu je rozdělena na tři obrazovky a k jejímu ovládání se používá zařízení gamepad, který je nutné držet po celou dobu. Hlavním grafickým ovládacím prvkem je kontextové menu, které se dá vyvolat pomocí tlačítka menu. Pro pohyb v menu se používají směrové šipky a výběr položek se provádí pravým tlačítkem trigger. Levým tlačítkem trigger se opouští menu nebo aktuální obrazovka. Na hlavní obrazovce lze vidět mapu a bezpilotní pro-

Obrázek 5.1: Popis ovládacích tlačítek na gamepad

středky. Výběr konkrétního prostředku lze provést zaměřením pohledu na něj a zmáčknutím pravého tlačítka pro potvrzení nebo volbou z kontextového menu. Výběr prostředku vás přepne na obrazovku zobrazující telemetrické údaje prostředku. Pomocí kontextového menu pak můžete tvořit a ovládat provádění jeho misí. Vytváření bodů mise se provádí zaměřením pohledu na konkrétní místo mapy a zmáčknutím potvrzovacího tlačítka. Z kontextového menu také spustíte manuální ovládání výběrem položky „Direct control“. Manuální ovládání směru se provádí levou pákou. Plyn se ovládá pravým analogovým tlačítkem a brzda levým.

5.1.2 Uživatelské testy

K otestování funkčnosti aplikace byli vybráni tři absolventi ČVUT - A, B, C. Osoba A má oborové zaměření strojírenství a osoba B stavebnictví. Tito lidé byli vybráni, protože bylo třeba ověřit, že ovládání je dostatečně snadné i pro osoby bez hluboké znalosti počítačů. Jako kontrolní byla vybrána osoba C se zaměřením na programování a správu počítačů. Uživatelé byli před testem seznámeni se systémem a jeho ovládáním. Za úkol jim bylo dáno splnit seznam úkolů uvedený v odstavci 5.1.2.1, které otestují funkčnost vytvořeného systému.

5.1.2.1 Seznam úkolů

1. Prosím nalezněte UAS v prostoru a na mapě.
2. Vyvolejte menu a zvolte UAS.
3. Zjistěte GPS pozici UAS.
4. Zapněte ruční dálkové ovládání.
5. Proleťte se s kvadrukoptérou.
6. Ukončete manuální ovládání.
7. Zobrazte si mapu pro plánování mise.
8. Naplánujte trasu.
9. Spusťte naplánovanou trasu.
10. Vraťte se na výchozí obrazovku.

5.1.3 Záznam testu

Během testování byli uživatelé pozorováni, aby byla zhodnocena jejich námaha při ovládání programu. Po ukončení zadaných úkolů byl s každým subjektem proveden rozhovor, ve kterém byly shrnuty klady a zápory výsledného programu. Výsledky rozhovorů a pozorování byly zaznamenány pro vyhodnocení:

A Ze začátku měl uživatel problémy se přizpůsobit ovládání na gamepadu a všechny své ze začátku operace opticky verifikoval. Nalezení prostředku a jeho zvolení proběhlo bez problému. Se získáním informací o prostředku neměl problém, přestože si uživatel stěžoval na čitelnost popisku. Manuální ovládání hodnotil uživatel jako neuspokojivé vzhledem k jeho virtuální simulaci. Při plánování trasy měl uživatel problém s přesným zaměřováním bodů.

B Stejně jako uživatel A, tento uživatel vykazoval problémy se zvykáním si na gamepad, které probíhaly do konce testu. Zobrazované mapy se uživateli zdály „pomuchlané“ a složité pro orientaci. Uživatel také měl problémy v rozlišováním mezi jednotlivými položkami v menu. Již po krátké chvíli si také začal stěžovat na otlačování způsobené brýlemi. Bez asistence by asi nebyl schopen daný seznam úkolů dokončit.

C Dle očekávání tento uživatel měl nejmenší problémy s ovládáním programu. Chvíli mu však trvalo si zvyknout na systém ovládání. Uživatel poukázal na viditelný rastr obrazovky a také na „rozmazávání“ obrazu při rychlejších pohybech. Množství podávaných informací se mu zdálo jako nízké, přesto však označil informační head-up display u manuálního ovládání jako „přeplácaný“.

5.1.4 Výsledku testu

Při testování s uživateli byla pravděpodobně největším problémem jejich nezkušenost s augmentovanou nebo i virtuální realitou. Přestože při vývoji byla virtuální simulace velice užitečná, při testování s uživateli se projevila jako nedostatečná. Většina problémů hlášených uživateli se vztahuje k použitému hardware. Viditelný rastr obrazovky nebo otlačování brýlí jsou minoritní. Nijak nebrání používání systému a lze je jednoduše odstranit použitím lepšího hardware. Počáteční zmatenost v augmentované realitě nebo chybějící svalová paměť se u většiny uživatelů po krátké době odpadla a další ovládání bylo bezproblémové. Využití pohybu k zadávání bodů se ukázalo být velice rychlé a užitečné, avšak jeho přesnost nebyla ideální. Ovládání pomocí kontextového menu se ukázalo být účinné, patrně díky omezenému množství položek.

5.2 Testování na hardware

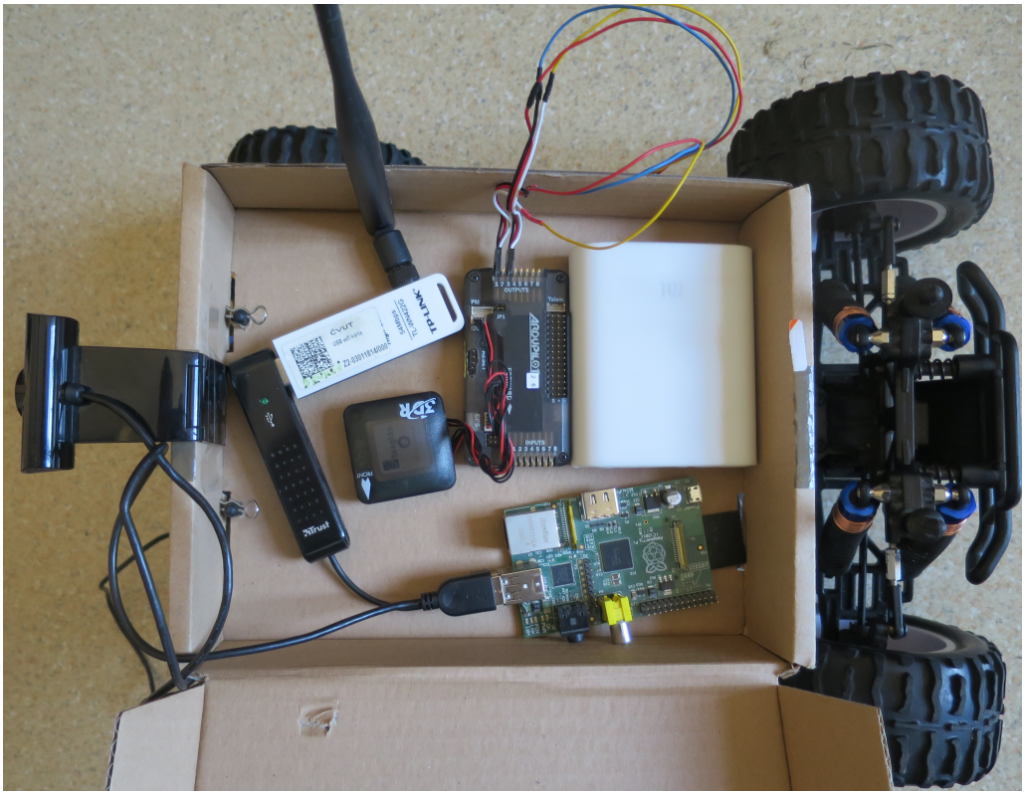
Pro ověření funkčnosti systému v reálných podmínkách bylo vytvořeno bezpilotní vozidlo. Jako základ byl použita pozemní robotická platforma ovládaná ArduPilotem s GPS přijímačem. Ardupilot byl připojen k Raspberry Pi, na kterém běžela aplikace LookingGlassUAS. K Raspberry Pi byla dále připojena webová kamera Creative Live! Cam Sync HD a Wi-Fi adaptér TP-LINK TL-WN422G. Podvozek byl napájen samostatným NiMH akumulátorem zatímco zbytek elektroniky byl napájen Xiaomi Power Bank. Testování probíhalo v areálu FEL ČVUT na Karlově náměstí.

Během testu byly ověřeny hlavní vlastnosti použitého software. Ověřovány byly možnosti lokalizace vozidla pomocí dat z GPS a jeho zobrazení na virtuální mapě. Během testu se také vytvářeli mize pro vozidlo. Nakonec se také otestovalo ruční dálkové ovládání a přenos živého videa z vozidla.

Obrázek 5.2: Ukázka průběhu testování systému na hardwarovém bezpilotním prostředku

5.2.1 Výsledky testu

Testování fyzického bezpilotního prostředku potvrdilo, že systém je reálně použitelný a relativně jednoduše nasaditelný. Problémy se objevily při video přenosu, kdy starý Wi-Fi adaptér a pouze integrovaná anténa mobilního telefonu neposkytují dostatečně dobrý signál. Pro zlepšení dosahu by bylo třeba použít externí anténu pro mobilní telefon a moderní Wi-Fi adaptér s více anténami. Dálkové manuální ovládání vyžadovalo chvíli zvykání si na vlastnosti vozidla a omezený zorný úhel daný webovou kamerou. Přesnost GPS dle předpokladu nebyla ideální, ale postačovala na určení pozice vozidla na virtuální mapě.



Obrázek 5.3: Výbava pozemního bezpilotního prostředeku použitého k testování

Závěr

V této práci jsem vytvořil implementaci systému pro ovládání bezpilotních prostředků s využitím augmentované reality. Výsledkem práce je aplikace pro mobilní telefony jménem Looking Glass a aplikace pro počítač ovládající bezpilotní prostředek jménem Looking Glass UAS. Spolu s doporučeným hardware tyto dvě aplikace umožňují ovládání několika bezpilotních prostředků jedním operátorem najednou. Operátorovi systém umožňuje mít přehled o pozici prostředků, plánování jejich misí a případně jejich ruční dálkové ovládání spolu s živým video přenosem.

Při vývoji systému jsem vytvořil několik unikátních modulů, které mohou být použity i v jiných aplikacích. Mezi ně patří rychlé promítání obrazu kamery telefonu na obrazovku, vykreslovací systém pro kompenzování distorze čoček v brýlích, multiplatformní rozhraní pro přístup ke gamepadu a implementace načítání a zobrazování map od Mapy.cz.

Výsledný systém jsem nechal otestovat několika uživateli pomocí virtuálních bezpilotních prostředků. Z výsledků rozhovorů a pozorování jsem vyhodnotil použitelnost systému a vyvodil, které věci by bylo třeba vylepšit. Kromě simulací jsem provedl i pokus s pozemním bezpilotním prostředkem, na kterém jsem ověřil použitelnost systému v reálném prostředí.

Pro další vývoj systému bych doporučoval inovovat hardware, zejména zobrazovací. Ovládání systému by také šlo obohatit o hlasové příkazy. Zpětná vazba by kromě grafiky mohla být doplněna zvukem a haptikou. Stávající simulaci bezpilotních prostředků lze rozšířit na plně virtuální systém pro výuku řízení. Kromě kooperace prostředků by bylo možné přidat i podporu kooperace operátorů.

Literatura

[1] *Comparison of mobile operating systems*. [cit. 2015-12-29]. Dostupné z: https://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems

[2] *MAVLink Micro Air Vehicle Communication Protocol*. [cit. 2015-12-29]. Dostupné z: http://qgroundcontrol.org/mavlink/start

[3] Aaron Hall, MBA: *VideoCore IV BCM2835 Overview*. [cit. 2016-1-4]. Dostupné z: https://github.com/hermanhermitage/videocoreiv/wiki/VideoCore-IV---BCM2835-Overview

[4] Andreas Parsch: *Boeing AGM-86 ALCM*. [cit. 2015-12-16]. Dostupné z: http://www.designation-systems.net/dusrm/m-86.html

[5] Ardupilot: *ArduPilot | Open source autopilot*. [cit. 2015-12-25]. Dostupné z: http://ardupilot.com/

[6] Christian de Looper: *The complete list of self-driving cars in development*. [cit. 2016-1-5]. Dostupné z: http://www.techradar.com/news/car-tech/the-complete-list-of-self-driving-cars-in-development-1306382

[7] Draganfly Innovations Inc: *A Short History of Unmanned Aerial Vehicles (UAVs)*. [cit. 2015-12-16]. Dostupné z: http://www.draganfly.com/news/2009/03/04/a-short-history-of-unmanned-aerial-vehicles-uavs/

[8] Emmanuel Dupuy: *Java Decompiler*. [cit. 2015-11-08]. Dostupné z: http://jd.benow.ca/

[9] Epson: *Moverio Smart Glasses*. [cit. 2015-02-04]. Dostupné z: http://www.epson.com/cgi-bin/Store/jsp/Landing/moverio-bt-200-smart-glasses.do

[10] Google: *CaptureRequest*. [cit. 2015-02-05]. Dostupné z: https://developer.android.com/reference/android/hardware/camera2/CaptureRequest.html#SENSOR_FRAME_DURATION

[11] Google: *Github cardboard-java*. [cit. 2014-9-06]. Dostupné z: https://github.com/googlesamples/cardboard-java/blob/master/CardboardSample/libs/cardboard.jar

[12] Gorry Fairhurst: *Unicast, Broadcast, and Multicast*. [cit. 2016-1-2]. Dostupné z: http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/uni-b-mcast.html

[13] H@Lu$k@: *Barevné vnímání oka*. [cit. 2015-12-29]. Dostupné z: http://lasery.kvalitne.cz/index.php?text=31-barevne-vnimani-oka

[14] http://oculusrift-blog.com: *John Carmack's Delivers Some Home Truths On Latency*. [cit. 2015-02-06]. Dostupné z: http://oculusrift-blog.com/john-carmacks-message-of-latency/682/

[15] Leap Motion: *Leap Motion*. [cit. 2015-12-16]. Dostupné z: https://www.leapmotion.com/

[16] Lynne E. Parker: *Current Research in Multi-Robot Systems*. [cit. 2015-12-15]. Dostupné z: http://web.eecs.utk.edu/~leparker/publications/JALR03.pdf

[17] Magic Eye, Inc.: *Stereo Vision starts with two views!* [cit. 2015-11-07]. Dostupné z: http://www.vision3d.com/stereo.html

[18] Master Sergeant Steve Horton: *MQ-1 Predator controls 2007-08-07*. [cit. 2015-12-10]. Dostupné z: https://commons.wikimedia.org/wiki/File:MQ-1_Predator_controls_2007-08-07.jpg

[19] Microsoft: *Cortana interactions*. [cit. 2015-12-16]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/apps/mt185598.aspx

[20] Microsoft: *Microsoft HoloLens*. [cit. 2015-12-15]. Dostupné z: https://www.microsoft.com/microsoft-hololens/en-us

[21] Milan Rollo: *Právní aspekty provozu UAV v ČR a ukázka bezpilotních prostředků*. [cit. 2015-12-15]. Dostupné z: http://webdav.agents.fel.cvut.cz/data/teaching/bep/BEP11_2014.pdf

[22] Pajs: *Cocka vada zkresleni*. [cit. 2015-12-10]. Dostupné z: https://cs.wikipedia.org/wiki/%C4%8Co%C4%8Dka_%28optika%29#/media/File:Cocka_vada_zkresleni.svg

[23] Prodromos-vasileios Mekikis: *Will we need a driving license to drive a car in ten years?* [cit. 2015-12-15]. Dostupné z: https://ec.europa.eu/digital-agenda/en/content/will-we-need-driving-license-drive-car-ten-years

[24] The Qt Company: *Qt*. [cit. 2015-12-29]. Dostupné z: http://www.qt.io/

[25] R. Even, T. K. R. J., Y.-K. Wang: RTP Payload Format for H.264 Video. IETF RFC 6184, 11 2011.

[26] Raspberry Pi Foundation: *Raspberry Pi*. [cit. 2015-12-28]. Dostupné z: https://www.raspberrypi.org

[27] Royal Navy Submarine Museum: *Curator's Choice -Whitehead torpedo*. [cit. 2015-12-16]. Dostupné z: http://www.submarine-museum.co.uk/what-we-have/our-favourite-objects/whitehead-torpedo

[28] Sean Hollister: *This Is How Valve's Amazing Lighthouse Tracking Technology Works*. [cit. 2015-12-17]. Dostupné z: http://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768

[29] Seznam.cz: *API Mapy.cz*. [cit. 2015-12-28]. Dostupné z: https://api.mapy.cz/

[30] Sir Howard Grubb, B. G. T., Dawson Arthur Trevor: *Improvements in Sighting Devices for Guns*. Dostupné z: http://www.directorypatent.com/GB/190022127-a.html

[31] SKYbrary: *Head Up Display*. [cit. 2015-12-30]. Dostupné z: http://www.skybrary.aero/index.php/Head_Up_Display

[32] Steven Dutch: *Converting UTM to Latitude and Longitude (Or Vice Versa)*. [cit. 2016-1-3]. Dostupné z: http://www.uwgb.edu/dutchs/usefuldata/utmformulas.htm

[33] Streaming Media: *The Great UHD Codec Debate: Google's VP9 Vs. HEVC/H.265*. [cit. 2016-1-4]. Dostupné z: http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/The-Great-UHD-Codec-Debate-Googles-VP9-Vs.-HEVC-H.265-103577.aspx

[34] Sutherland, I. E.: *A head-mounted three dimensional display*. [cit. 2015-12-15]. Dostupné z: http://design.osu.edu/carlson/history/PDFs/p757-sutherland.pdf

[35] Trond Jacobsen: *ZEVS, THE RUSSIAN 82 Hz ELF TRANSMITTER*. [cit. 2016-1-5]. Dostupné z: http://www.vlf.it/zevs/zevs.htm

[36] University of Pennsylvania: *Quadrotors Come to TED*. [cit. 2015-02-04]. Dostupné z: http://www.upenn.edu/spotlights/penn-quadrotors-ted

[37] US Goverment: *International Traffic In Arms Regulations*. [cit. 2015-12-15]. Dostupné z: http://fas.org/spp/starwars/offdocs/itar/p121.htm

Příloha A

Seznam použitých zkratek

UAS Unmanned aerial system

UGV Unmanned ground vehicle

UAV Unmanned aerial vehicle

GUI Graphical user interface

FPS Frames per second

NDK Native development kit

GLSL OpenGL Shading Language

API Application programming interface

GPU Graphics processing unit

CPU Central processing unit

IDE Integrated development environment

QML Qt Meta Language

JNI Java Native Interface

IO Input / Output

UDP User Datagram Protocol

RTP Real-time Transport Protocol

RTCP RTP Control Protocol

ELF Extremely low frequency

MEO Medium Earth orbit

HMD Head mouted display

IMU Inertial measurement unit

MTU Maximum transmission unit

WGS84 World Geodetic System 1984

UTM Universal Transverse Mercator

GNSS Global Navigation Satellite System

Příloha B

Instalační příručka

B.1 Android (Lollipop)

Na svém telefonu si v nastavení změňte povolení pro instalaci aplikací z neznámých zdrojů do polohy povoleno. Nakopírujte si do telefonu instalační balíček `LookingGlass.apk` z adresáře `/exe` na CD. V telefonu proveďte jeho instalaci.

B.2 Desktopový počítač s GNU/Linux (Ubuntu 15.10)

Doinstalujte si potřebné balíčky pomocí příkazu B.1. Program pro augmentované ovládání prostředků LookingGlass naleznete v adresáři `/exe` na CD. Ve stejném adresáři se také nachází program pro bezpilotní prostředek `LookingGlassUAS_x86`.

Zdrojový kód B.1: Instalační příkaz

```
sudo apt-get install libqt5qml5 libqt5core5a \
        libqt5location5 libqt5multimedia5 \
        libavutil-ffmpeg54 libqt5positioning5 \
        libgstreamer1.0-0
```

B.3 Raspberry Pi (Raspbian Jessie)

Doinstalujte si potřebné balíčky pomocí příkazu B.2. V adresáři `/exe` na CD najděte spustitelný program pro bezpilotní prostředek `LookingGlassUAS_arm`.

Pro lepší komfort upravte systém dle B.3 a B.4.

Zdrojový kód B.2: Instalační příkaz

```
sudo apt-get install libgstreamer1.0-0 gstreamer1.0-omx
```

Zdrojový kód B.3: Obsah /etc/rc.local, spouštějící se automaticky po spuštění systému program LookingGlassUAS

```
#!/bin/sh -e-
sleep 30
/path_to_the_executable/LookingGlassUAS_arm -r &
exit 0
```

Zdrojový kód B.4: Obsah /etc/wpa_supplicant/wpa_supplicant.conf, nastavující parametry Wi-Fi sítě

```
network={
    ssid="LookinGlass"
    psk="lookingglass"
}
```

Příloha C

Obsah CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `Ovládání týmu robotických prostředků s využitím augmentované reality.pdf` ..... text práce ve formátu PDF
  - `Ovládání týmu robotických prostředků s využitím augmentované reality.ps` ..... text práce ve formátu PS