Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Experimentálne prostredie pre interakciu človeka s humanoidným robotom iCub

Bakalárska práca

2022
Sabína Samporová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Experimentálne prostredie pre interakciu človeka s humanoidným robotom iCub

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	RNDr. Kristína Malinovská, PhD.

Bratislava, 2022
Sabína Samporová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Sabína Samporová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Prepojenie virtuálnej reality so simulovaným robotom ICub
Simulated ICub robot in virtual reality

Anotácia: ICub[1] je stredne veľký humanoidný robot, ktorý svojím vzhľadom a dispozíciami ako je napríklad počet stupňov voľnosti pripomína malé dieťa. Bol vyvinutý v Európe pre účely výskumu v kognitívnej robotike a od svojho vzniku pred skoro 10 rokmi sa stal veľmi populárny a vedcami často používaný aj vďaka jeho simulovanej podobe [2]. V novátorskom výskume v oblasti interakcie človeka s robotom (human-robot interaction, HRI) sa venujeme možnosti interakcie a trénovania robotov pomocou ľudí vo VR [3]. V pilotnej časti výskumu skúmame vnímanie robota človekom v 3 scenároch - na živo, vo VR a na monitore. Všetky tri scenáre zahŕňajú jednoduchú nekontaktnú interakciu medzi robotom a človekom [5], pričom rovnaký program bude použitý - a musí byť teda aj otestovaný - pre všetky tri platformy.

Cieľ: Preštudujte potrebnú literatúru o simulátore ICub, robotickej platforme YARP, ktorú používa [4].

Literatúra:

[1] Metta G, Sandini G, Vernon D, Natale L, Nori F. The iCub humanoid robot: an open platform for research in embodied cognition. In Proceedings of the 8th workshop on performance metrics for intelligent systems 2008 Aug 19 (pp. 50-56). ACM.

[2] Tikhanoff V, Cangelosi A, Fitzpatrick P, Metta G, Natale L, Nori F. An open-source simulator for cognitive robotics research: the prototype of the iCub humanoid robot simulator. In Proceedings of the 8th workshop on performance metrics for intelligent systems 2008 Aug 19 (pp. 57-61). ACM.

[3] Krupke, Dennis, Sebastian Starke, Lasse Einig, J. Zhang, and F. Steinicke. "Prototyping of immersive HRI scenarios." In Human-Centric Robotics: Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, pp. 537-544. 2018.

[4] http://www.yarp.it/

[5] Wiese, Eva, Patrick P. Weis, and Daniel M. Lofaro. "Embodied social robots trigger gaze following in real-time HRI." In 2018 15th International Conference on Ubiquitous Robots (UR), pp. 477-482. IEEE, 2018.

Poďakovanie: todo

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

TODO

Prečo skúmať robota vo VR

treba napisat ze motivaciou pre vytvorenie prostredia je dany experiment, a ze sucast prace je aj samotne meranie experimentu

ze spolupracujem s Kassandrou

nasa hypoteza

Kapitola 1

Východiská práce

todo finalne preformulovatV tejto kapitole opisujeme stav problematiky v oblasti human-robot interaction do ktorej bude náš experiment prispievať. Do experimetu vstupujú otázky stelesnenia, formy stelesnenia a prítomnosti umelo inteligentného agenta a vplyvu týchto vlastností na vnímanie robota človekom. Ďalej sú uvedené všetky technológie, ktoré boli v práci, prípadne pri vyvíjaní prostredia na interakciu človeka s robotom využité.

1.1 Teoretické východiská

1.1.1 Interakcia človeka a robota

Interakcia človeka a robota (Human–robot interaction, alebo aj HRI) je oblasť, ktorá prepája myšlienky a poznatky z niekoľkých disciplín. Spája strojárstvo, informatiku, robotiku, psychológiu, sociológiu ale napríklad aj dizajn. Každá z týchto disciplín má v HRI isté zastúpenie. Interakcia človeka a robota sa venuje štúdiu vzájomného pôsobenia a komunikácie človeka a robota. [1]

Rozdielom medzi HRI a robotikou je, že zatiaľ čo robotika sa zaoberá vyvinutím fyzického robota a jeho manipuláciou s fyzickým svetom, HRI sa sústreďuje na spôsoby ako robot interaguje s ľudmi v tomto svete. [1]

Interakcia človeka a robota sa zameriava na vyvíjanie takých robotov, ktorí sú schopní interagovať s ľuďmi v rôznych každodenných prostrediach. Skúmanie interakcia človeka s robotom je považované za veľmi dôležité, pretože umelo inteligentní agenti sa čoraz viac stávajú súčasťou našej spoločnosti. Napríklad roboti určení na opatrovanie a poskytovanie starostlivosti poskytujú dohľad a sociálnu interakciu pre deti a starších ľudí, robotický asistenti pre výučbu pomáhajú učiteľom v školských triedach, asistenční roboti pomáhajú aj pacientom s ich stravovacím a terapeutickým režimom [2]. Nedávno sa aj na Slovensku objavil prvý robotický čašník, BellaBot.

1.1.2 Skúmané vlastnosti robota interagujúceho s človekom

Stelesnenie

Za pojmom stelesnenie (Embodiment) sa skrýva viac než to,že robot je len počítač na nohách alebo kolesách, teda že má vymodelované telo. Stelesnenie sa týka ako hardvéru, tak aj softvéru, pretože softvér je veľmi dôležitý pri tom, aký dojem urobí robot na ľudí a ako pôsobivo s nimi vie interagovať. Stelesnenie teda kladie fyzické obmedzenia na možnosti robota vo fyzickom svete a zároveň obmedzenia možností interakcie s ľuďmi v tomto svete.

Ak sa robot fyzicky dostatočne podobá na človeka, je pre ľudí jednoduchšie s ním komunikovať tak ako by to robili s človekom. Potom aj podobne ľahko aplikujú skúsenosti z medziľudskej interakcie do interakcie s robotom. [1]

O stelesnení môžeme uvažovať v dvoch formách, t.j. o fyzickom stelesnení a digitálnom stelesnení. Fyzicky stelesnený robot - fyzický agent - znamená, že robot má motory a je zložený z kovových, či plastových súčastí. Oproti tomu, digitálne stelesnený robot alebo virtuálny agent označuje animovaný renderovaný objekt vytvorený použitím počítačovej grafiky.

Stelesnenie je široko diskutovaná a študovaná otázka v oblasti kognitívnej vedy. Často kladené otázky v súvislosti so stelesnením sú, ako stelesnenie robota vplýva na naše myšlienkové procesy, ako na náš jazyk a vyjadrovanie a ako na naše správanie. Výsledkom experimentu na ktorý bude naše prostredie využité by malo byť skúmanie vplyvu rôznych foriem stelesnenia robota na interakciu človeka s robotom.

Prítomnosť

Zatiaľ čo stelesnenie nutne ukazuje na agenta a jeho vzťah k okolitému svetu, prítomnosť, teda presence, hovorí o prítomnosti robota vzhľadom k iným ľuďom. V tomto prípade teda vzhľadom k účastníkom experimentu.[3]

Pri prítomnosti opäť rozlišujeme podobné formy. Ide o fyzickú prítomnosť a digitálnu prítomnosť. Fyzická prítomnosť vyžaduje, aby robot a osoba boli vo fyzickej blízkosti, napríklad v rovnakej miestnosti. Fyzickú prítomnosť nazývame aj copresence a fyzicky prítomného robota nazývame copresent robot.

Digitálna prítomnosť označuje, že agent síce nie je vo fyzickej blízkosti človeka, ale je v jeho elektronickej blízkosti. Teda, že človek môže vnímať digitálneho robota a to je mu umožnené nejakým zariadením. Digitálna prítomnosť je označovaná ako telecopresence a digitálne prítomného robota nazývame telepresent robot.

Teda ak je agent v rovnakom fyzickom priestore ako človek, napríklad v rovnakej miestnosti, ide o copresent agenta. Ak je ale premietaný na obrazovke cez živý video prenos je telepresent agentom.

1.1.3 Doterajšie výsledky skúmania stelesnenia a prítomnosti

Problematika významu fyzickej alebo digitálnej prítomnosti a stelesnenia skúmal Li [3].

Výsledkom niekoľkých desiatok experimentov bolo, zodpovedanie na 3 základné položené otázky. Reagujú ľudia na copresent robota a na telepresent robota rozdielne? Reagujú ľudia na telepresent robota rozdielne ako na simulovaného agenta s veľmi podobným vzhľadom? Reagujú ľudia na copresent robota rozdielne ako na simulvaného agenta s veľmi podobným vzhľadom a správaním?

Z dát, ktoré zozbierali a vyhodnotili vyplýva, že väčšina ľudí reaguje priaznivejšie na copresent robota a menej na telepresent robota. Ďalej sa ukázalo, že fyzické stelesnenie nemalo vplyv na účastníkove reakcie. Napokon, väčšina ľudí reaguje priaznivejšie na copresent robota a menej na virtual agenta.[3]

1.1.4 Sociálne podnety

Premýšľanie o vnútornom stave človeka s ktorým interagujeme je automatické. Výskumy však preukázali, že pri interakcii človeka s robotom je potrebné aby robot vykazoval isté znaky sociálneho správania na to, aby ho človek vnímal ako konajúceho tvora a veril, že dokáže plánovať, uskutočňovať akcie a myslieť. Ak to chceme dosiahnuť, správanie robota sa musí podobať na ľudské, čo je spojené nie len s jeho správaním, ale aj s jeho výzorom. Robotovo stelesnenie teda musí mať ľudské črty.

Keď je už robot vnímaný ako konajúca bytosť, stúpa sociálna relevantnosť akcií ktoré vykoná, čo pozitívne ovplyvní jeho postavenie v interakcii človeka s robotom. Pohyb robotovych očí, prípadné prirodzené sledovanie očí človeka, je jeden zo sociálnych podnetov. Wiese, Weis a Lofaro [4] dokázali, že pri interaktívnej komunikácii ľudia pripisujú robotovmu pohľadu sociálnu relevanciu a robota vnímajú ako sociálneho aktéra.

1.1.5 Nasledovanie pohľadu

Ukazuje sa, že v medziľudskej komunikácii má neverbálne správanie, ktorým je napríklad aj nasledovanie pohľadu, či gestikulácia, istú úlohu a dôležitosť. Sledovanie pohľadu je zvlášť dôležitý neverbálny signál, oproti napríklad ukazovania alebo postoju tela. Psychologické výskumy ukazujú, že oči sú istým spôsobom špeciálny kognitívny stimul, a ľudský mozog ich interpretuje unikátnymi spôsobmi. [5]

Pohľad dokáže zaujať pozornosť inej osoby. Tento proces využitia pohybu očí ako informácia o tom, kam nasmerovať našu pozornosť je nazývaný gaze cueing. Efekty gaze cueing boli študované na ľuďoch, ale aj na iných primátoch. [5]

Pohyb očí hrá dôležitú rolu aj pri komunikácii o prostredí v ktorom sa nachádzame. Keď ľudia referujú na objekty v ich blízkosti, často sa na objekt pozrú a zafixujú svoj

pohľad na daný objekt približne jednu sekundu pred tým, ako ho pomenujú, alebo s ním začnú manipulovať. Ľudia sú veľmi dobrí v identifikovaní objektu na ktorý sa ich partner pozerá a vedia túto informáciu využiť na predikovanie čo sa im partner chystá povedať. [5]

1.1.6 Skúmanie efektu gaze cuing v interakcii človeka a robota

V už spomenutom experimente z George Manson University [4] skúmali interakciu človeka s fyzicky stelesneným robotom, ktorý pohybuje očami a hlavou do ľavej alebo pravej strany. Okrem robota sú súčasťou experimentu dve lampy. Jedna na ľavej a druhá na pravej strane. Po robotovom pohybe zasvieti jedna z lámp a účastník má na zasvietenie čo najrýchlejšie zareagovať stlačením tlačidla na príslušnej strane. Tento experiment preukázal gaze cuing pri interakcii človeka s robotom vo fyzickej forme stelesnenia a to tak, že človek bol schopný reagovať na zasvietenie lampy rýchlejšie, ak sa robot pozeral na lampu ktorá zasvietila. Ak sa pozeral na opačnú, reakcia bola v priemere pomalšia. Tým sa preukázalo, že človek vníma pohľad robota, prípadne ho nasleduje.

Naším cieľom bolo tento experiment rozšíriť o digitálnu a neskôr aj o viruálnu formu a tým odtestovať, ako vplýva forma stelesnenia robota na reakcie človeka. A teda v ktorej forme stelesnenia je efekt nasledovania pohľadu najsilnejší.

1.2 Technológie

Pre experiment bolo nutné vytvoriť program na meranie reakčného času s rôznymi formami stelesnenia robota. Návrh a vyvíjanie prostredia si vyžadovalo množstvo rôznych technológií.

1.2.1 Robot iCub

Robot iCub je humanoidný robot určený predovšetkým na výskumné účely, najmä na návrh a testovanie algoritmov umelej inteligencie ktoré vyžadujú fyzicky stelesneného robota, ktorý sa svojou stavbou tela a rozsahom pohybov podobá človeku. [6] ICub má približne 1 meter, váži 22 kilogramov a svojimi pohybovými schopnosťami sa najviac podobá na tri a pol ročné dieťa. Je vytvorený ako open project, a celý softvér pre prácu s robotom je open source. Vyvinutý bol na Italian Institute of Technology, Genoa. ICub má viac než 50 kĺbov, čo umožňuje vykonávanie veľmi rôznorodých pohybov. 16 kĺbov dokážeme ovládať na pravej ruke, ďalších 16 na ľavej, po 6 na oboch nohách, 6 na hlave a 3 na torze. [7]

Pre náš experiment bolo dôležité, aby robot pôsobil čo najprirodzenejšie a čo možno

najviac sa jeho pohyby podobali ľudským. ICuba sme vybrali nielen preto, lebo sa podobá na človeka, ale najmä preto, lebo má schopnosť hýbať hlavou, a ako jeden z mála podobne veľkých humanoidov dokáže hýbať očami a žmurkať. Z pohľadu kinematiky ide o aktuálne nejkomplexnejšieho humanoidného robota, aký bol nadizajnovaný. [8]

Fyzický robot iCub aj jeho softvér sú založené na platforme YARP.

Obr. 1.1: Fyzický robot iCub

1.2.2 YARP

YARP je open-source nástroj pre real-time aplikácie, ktoré sú výpočtovo náročné a vyžadujú prístup k rôznorodým a meniacim sa hardvérovým zariadeniam. [9]

Ide o skupinu knižníc, ktoré podporujú modularitu abstrahovaním dvoch najčastejších komplikácií v robotike : modularitu v algoritmoch a modularitu v prístupe k hardvéru. [10]

V našom experimente využívame najmä druhú vlastnosť. Keďže program využívame na rôznych formách robota. Samotné jadro programu na ovládanie pohybov robota je

vo všetkých formách čo najrovnakejšie a YARP zabezpečí, že aj napriek tomu, že sa zmenia hardvérové zariadenia na ktoré sa aplikuje, výsledný efekt bude v každej forme rovnaký. Takúto abstrakciu v praxi vykonáme pomocou YARP name servera.

1.2.3 YARP name server

YARP name server je nástroj ktorý umožňuje zjednodušené využitie YARP knižníc. YARP name server vytvára komunikačné kanály, resp. spojenia medzi pomenovanými entitami. Spojenia sa nazývajú porty. Každý port má unikátne meno, napr. pre ovládanie kĺbov ľavej nohy robota icub bude existovať port /icub/leg/right. Name server si pamätá spojenie s každým portom. Potom ak poznáme meno portu v YARP name serveri, je to jediné, čo potrebujeme na to, aby sme s portom komunikovali. Ak teda pošleme na niektorý port dáta, YARP name server nadviaže spojenie s fyzickým zariadením, prípadne iným resourcom a odošle mu dané informácie v správnom formáte.

Vo všeobecnosti, port môže poslať dáta ľubovoľnému množstvu iných portov a rovnako prijať dáta od ľubovoľného množstva iných portov. Spojenia môžu byť ľubovoľne pridávané a odoberané a tvoria sieť spojení.[10]

1.2.4 YARP motor GUI

Yarpmotorgui je program určený na ovládanie kĺbov robota, prostredníctvom YARP. Takto môžeme jednoducho testovať vychýlenie kĺbu do nejakej polohy pri návrh sekvencie pohybov robota. YARP motor GUI ponúka grafické používateľské rozhranie, kde je pre kazdý stupeň voľnosti každého kĺbu možné dynamicky meniť jeho hodnotu, pričom sa bezprostredne po zmene hodnoty zmena prejaví, tým, že robot iCub vykoná pohyb.

1.2.5 Simulátor robota iCub - iCubSIM

Podľa oficiálnej stránky robota iCub [6] v súčastnosti existuje 42 vyrobených fyzického robotov iCub. Pre ľudí, ktorí nemajú prístup k fyzickému robotovi, prípadne pre testovanie programu predtým, než ho necháme vykonať samotným robotom, existuje softvér Simulátor robota iCub - iCubSIM. Simulátor je Open-Source a na komunikáciu využíva, podobne ako samotný iCub, platformu YARP. To znamená, že simulátor aj reálny robot majú spoločný interface na ovládanie pohybu robota a teda ak spustíme rovnaký program na ovládanie robota, tak správanie v simulátore bude veľmi dobre zodpovedať správania fyzického robota.

Obr. 1.2: Robot iCub v simulátore iCubSIM

1.2.6 Gazebo

Gazebo je dynamický 3D simulátor so schopnosťou presne a efektívne simulovať robotov v komplexných prostrediach. Typicky sa využíva na testovanie robotických algoritmov, prípadne na návrh robotov. Gazebo ponúka bohatú knižnicu modelov robotov a prostredí a možnosť importovať robota z SDF modelu. [11] Gazebo je možné prepojiť s YARPom, teda robota iCub môžeme v Gazebo simulátore ovládať opäť rovnakým kódom ako v Simulátore iCubSIM a ako fyzického robota.

1.2.7 URDF a SDF modely

Unified Robotic Description Format (URDF) je xml formát definovaný pre ROS(Robot Operation System), ktorý popisuje všetky komponenty, z ktorých sa robot skladá. Pre Gazebo je ale URDF nedostatočný, pretože mu chýba možnosť definovať niektoré vlastnosti, ktoré sú v 3D simulátore potrebné. Ide napríklad o polohu robota v rámci sveta v simulátore. Pomocou URDF nie je možné špecifikovať ani iné objekty ktoré sa v Gazebo simulátore môžu vyskytovať. To sú napríklad o svetlá, výškové mapy a podobne.

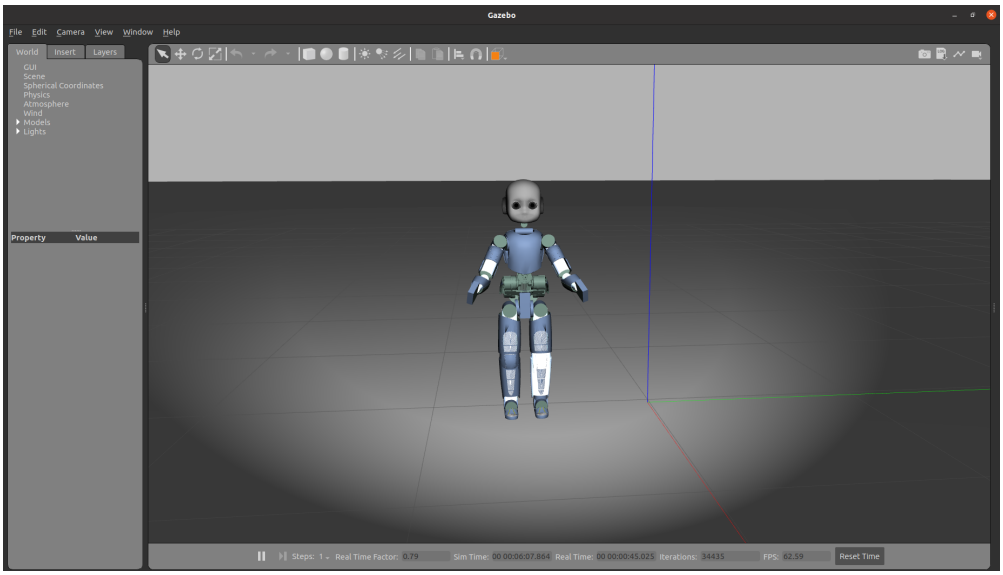
Ako riešenie prišlo Gazebo s novým formátom, ktorý nazvali Simulation Description Format(SDF). SDF je úplný popis všetkého, čo sa v Gazebe vyskytuje. Dá sa ním špecifikovať čokoľvek, od popisu sveta až po popis robotov. Je škálovateľný a modifikovanie elementov je jednoduché. Opäť ide o formát xml súborov. [11] Gazebo ponúka konverziu z URDF do SDF.

1.2.8 Modely robota iCub

K dispozícii sú dve dostupné alternatívy iCub URDF a SDF modelov. Staršie modely [12] obsahujú 5 rôznych modelov, z ktorých tri majú dostatočne podrobne vymodelované štruktúry materiálov, na to, aby sme ich mohli v experimente použiť. Zvyšným chýba hlava.

Nové modely [13] síce ponúkajú množstvo rôzne podrobných a upravených modelov, ale žiaden z nich nemá dostatočne podrobne vymodelované materiály na to, aby uveriteľne pripomínal skutočného robota.

Preto sme v experimente použili SDF model iCub fixed zo starších modelov. Tento má okrem vernej štruktúry aj pevne zafixované telo, teda v simulátore stojí a pri pohybe hlavy a očí ostáva stabilný, nespadne na zem vplyvom gravitácie v simulátore.



Obr. 1.3: SDF model robota iCub v Gazebo simulátore

1.2.9 Robotology-superbuild

Robotology-superbuild je infraštruktúra, ktorá zjednodušuje vývoj a využívanie open source softvéru vyvinutého v Italian Institute of Technology, ako súčasť projektu iCub. [14]

Je to najjednoduchší spôsob ako získať kompatibilné verzie vyššie spomenutých technológií ako je YARP, simulátor robota iCub, Gazebo a ďalšie.

1.2.10 C++

C++ je viacparadigmový programovací jazyk vyššej úrovne na všeobecné použitie, ktorý umožňuje pracovať aj s prostriedkami nízkej úrovne. Má statickú typovú kon-

trolu, podporuje procedurálne programovanie, dátovú abstrakciu, objektovo orientované programovanie, ale aj generické programovanie. [15]

C++ je primárny jazyk na prácu s YARPom, teda aj s fyzickým robotom, Simulátorom robota iCub a Gazebo simulátorom.

Z toho dôvodu je prostredie, ktoré sme vyvinuli, implementované v jazyku C++.

1.2.11 HTC Vive Pro

HTC Vive Pro je headset pre virtuálnu realitu vyvíjaný spoločnosťou Valve. Tento headset bol vytváraný na používanie v priestore, preto obsahuje dve základné stanice, ktoré sa umiestnia do dvoch protilahlých rohov miestnosti a vymedzia priestor 3×4 metre. Následne sa používateľ môže v tomto priestore pohybovať a stanice snímajú jeho polohu. K HTC Vive Pro patria aj dva haptické ovládače.

Obr. 1.4: Virtuálna realita HTC Vive Pro

1.2.12 Arduino Nano

Arduino je open-source platforma založená na princípe jednoducho ovládateľného hardvéru aj softvéru. Arduino dosky majú možnosť čítať vstupy z pripojených senzorov, spracovávať ich jednoduchou logikou a premieňať ich na výstupy napr. aktivovanie motorov, zapínanie LED diód a podobne. Správanie dosky je závislé od inštrukcií, programu poslaného na mikrokontrolér dosky. Program pre Arduino je písaný v programovacom jazyku Arduino, ktorý je založený na C++.

Arduino Nano je klasická doska založená na procesore ATmega328. Je napájaná cez USB, obsahuje niekoľko analógových pinov a digitálnych pinov, na ktoré vie dávať napätie $5V$. Výhodou je, že aj napriek množstvu pinov má pomerne malé rozmery a je praktická na používanie.

1.2.13 Arduino Ide

Na naprogramovanie inštrukcií pre Arduino a následný upload na dosku je vyvinutá open-source platforma, Arduino IDE, ktorá uľahčuje prácu s doskou. Arduino Ide dokáže pracovať so všetkými druhmi Arduino dosiek.

Kapitola 2

Špecifikácia psychologického experimentu

Táto kapitola uvádza podrobné informácie o experimente na ktorý sme využili vytvorený softvér, ako aj to, aké požiadavky na základe špecifikácie experimentu bolo potrebnú zahrnúť vo vytváranom prostredí. Forma experimentu, ktorý sme vykonávali nadväzuje na článok [4].

Experiment skúma vplyv formy stelesnenia robota na mieru nasledovania pohľadu tohto robota človekom. Experiment zbiera dáta pomocou zaznamenávania rýchlosti a správnosti reakcie osoby na stimul v podobe zasvietenia svetla. Cieľom experimentu bolo preskúmanie rôznych formiem stelesnenia robota - fyzicky stelesnený robot, digitálne stelesnený robot a virtuálne stelesnený robot. Teda skutočného robota, robota vizualizovaného na obrazovke počítača a robota zobrazeného vo virtuálnej realite.

2.1 Formy stelesnenia robota

2.1.1 Fyzická forma stelesnenia

V tejto forme má robot fyzické telo, zložené z fyzických kĺbov, pomocou ktorých vie interagovať. Ide teda o jedného zo 42 doteraz existujúcich fyzických robotov iCub. Robot sa nachádza v rovnakej miestnosti, ako účastník experimentu. Súčasťou experimentu sú aj fyzické lampy, ktoré je možné ovládať, zapínať a vypínať. Na odchytenie reakcie na stimul je využitá klávesnica.

2.1.2 Digitálna forma stelesnenia

Robot má telo vymodelované v počítačovom simulátore a pomocou simulovaného prostredia je zobrazený na obrazovke počítača. Účastník experimentu je v miestnosti s

obrazovkou na ktorej je iCub, okrem toho sa v miestnosti nachádzajú fyzické lampy a na zaznamenianie reakciu opäť používame klávesy na klávesnici.

2.1.3 Virtuálna forma stelesnenia

Virtuálne stelesnený robot má telo vymodelované v 3D simulátore. Virtuálna forma používa Náhlavný displej - zobrazovacie zariadenie, ktoré má malú optiku niekoľko centimetrov pred očami a zobrazuje premietaný obraz v 3D. Účastník experimentu má nasadený tento Náhlavný displej a pomocou neho sa premieta virtuálny robot iCub. Lampy v tejto forme nie sú fyzické, ale sú súčasťou premietanej scény. Reagovanie účastníka je opäť zaznamenávané pomocou kláves na klávesnici, prípadne pomocou haptických ovládačov pre virtuálnu realitu.

2.2 Priebeh experimentu

2.2.1 Priebeh jedného trialu

Jeden trial sa skladá zo 4 fáz.

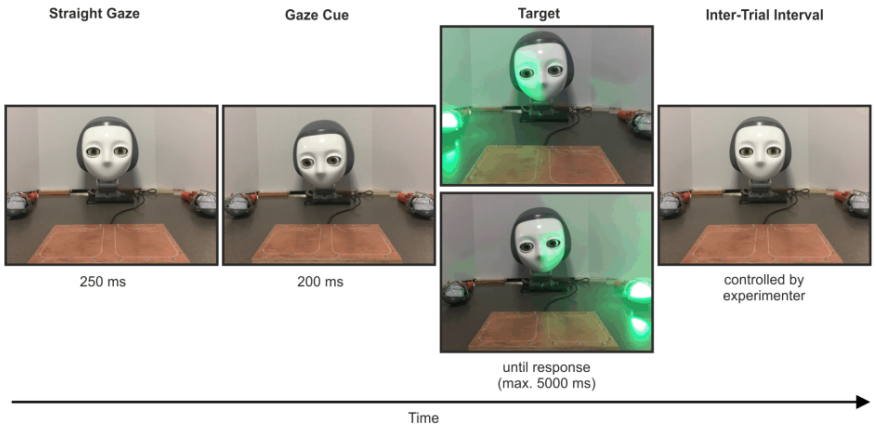
V prvej fáze sa robot pozerá priamo vpred, na miesto kde sedí účastník. Tým sa snaží nadviazať očný kontakt a zaujať účastníka. Po 250 milisekundách prejde do 2 fázy, kedy robot vykoná pohyb hlavy a očí buď doprava alebo doľava. Tento pohyb znamená upriamenie pohľadu na jedno zo svetiel. Ak sa hýbal napravo, tak na pravé svetlo, ak naľavo, tak na ľavé svetlo. Po 200 milisekundách, v 3 fáze sa zasvieti jedno zo svetiel, buď stranovo súhlasné alebo opačné. Vtedy má účastík čo najrýchlejšie zareagovať stlačením tlačidla pre príslušnú (ľavú alebo pravú) stranu, podľa toho, kde svieti svetlo. V poslednej fáze sa robot vráti do iniciálnej polohy, kde opäť nadväzuje očný kontakt a pozerá priamo na účastníka. Súčasťou každého trialu je meranie a zapisovanie reakčného času, t.j. času od zasvietenia lampy po stlačenie klávesy.

2.2.2 Konfigurácie experimentu

Každý trial má priradenú 1 zo špecifických konfigurácií, ktoré určujú smer pohybu robota iCub a lampu ktorá zasvieti. Jedna konfigurácia sa skladá z

- poradového číslo konfigurácie
- označenia strany, na ktorú ukáže robot
- označenia strany, na ktorej zasvieti lampa

V experimente sa vyskytujú 4 rôzne druhy konfigurácií:



Obr. 2.1: Zobrazenie experimentu [4]

- pohľad vľavo, lampa vľavo
- pohľad vpravo, lampa vpravo
- pohľad vľavo, lampa vpravo
- pohľad vpravo, lampa vľavo

2.2.3 Štruktúra experimentu

Experiment je zložený z 80 trialov na každého účastníka, v týchto je zastúpených po 20 z každého vyššiepopísaného druhu konfigurácií. Konfigurácie sú na začiatku preusporiadané do náhodného poradia.

Keď je účastík pripravený, spustí sa program ovládajúci robota, svetlá a meranie reakcií. Každý trial sa vykoná podľa konfigurácie ktorá mu prislúcha. Zakaždým sa do výstupného súboru uloží poradové číslo konfigurácie, samotná konfigurácia, rýchlosť a správnosť reakcie.

2.2.4 Úloha účastníka experimentu

Každý účastník absolvuje experiment iba s jednou z foriem robota.

Účastník experimentu sa nachádza v niektorom spoločnom prostredí s robotom. Vidí robota a dve lampy. Jednu na pravej strane od robota, druhú na ľavej. Účastník má v dosahu dve tlačidlá - ľavé a pravé. Úlohou účastníka je, po zasvietení niektorého zo svetiel, čo najrýchlejšie zareagovať tým, že stlačí tlačidlo. Ak zasvietilo pravé svetlo, má stlačiť tlačidlo napravo, ak ľavé, má reagovať stlačením ľavého tlačidla.

Účastník dostane informáciu, že robot sa bude pohybovať, ale nie je priamo vyzývaný, aby robota sledoval.

2.2.5 Pseudokód experimentu

Ako bolo uvedené v predchádzajúcich častiach špecifikácie, program pre meranie experimentu má niektoré význačné časti. Ide o špecifické časy medzi fázami, prechádzanie konfiguráciami, meranie reakčného času a zapisovanie nameraných výsledkov. Tieto časti môžeme zapísať vo forme nasledujúceho pseudokódu.

```
participantNumber = number from std input
configurations [] = array with 20 of each configuration types

random shuffle configurations

for confNumber in (0,1, ... 79)
{
    actConf = configurations[confNumber]

    wait 250 ms
    move robot head and eyes to actConf.moveSide

    wait 200 ms
    time0 = current time
    light up lamp in actConf.lampSide

    waiting for input
        keyPressed = character from std input

    reactionTime = current time - time0

    move robot head and eyes to initial position

    write (participantNumber, confNumber, actConf.moveSide,
    actConf.lampSide, keyPressed, reactionTime) to output file
}
```

2.3 Ciele experimentu

Cieľom experimentu je získať dáta a následne zanalyzovať, pri ktorej z foriem robota bude človek najviac ovplyvňovaný pohybom robota, teda v ktorej forme sa efekt gaze cuing prejaví najsilnejšie.

Očakávame, že z dát vyplynie, že keď sa robot zapozerá na svetlo, na základe efektu nasledovania pohľadu upriami účastník svoj zrak na rovnaké svetlo. Potom ak toto stranovo súhlasné svetlo zasvieti, reakcia účastníka by mala byť veľmi rýchla, keďže už vopred predpokladá zasvietenie tohto svetla a venuje mu pozornosť. Naopak, ak zasvieti stranovo nesúhlasné svetlo, účastníkovi by mala trvať reakcia dlhšie, pretože sa sústredí na iný objekt. Toto sa potvrdilo aj v pôvodnej verzii experimentu [4].

Ďalším skúmaným javom je práve forma stelesnenia a porovnanie vplyvu formy stelesnenia na mieru efektu gaze following. Naša hypotéza je, že pri fyzickej forme bude gaze following omnoho zretelnejšie než pri simulovanej a virtuálnej forme. Vychádzame z výsledkov výskumu [3], ktoré sú popísané v časti 1.1.3.

Kapitola 3

Návrh a výroba lámp na vytváranie svetelných stimulov

Kľúčovou časťou experimentu z ktorého vychádzame [4] je meranie rýchlosti reakcie na stimuly. V pôvodnom návrhu boli na vytváranie stimulov použité dve smart žiarovky TP-Link LB130. Keďže išlo o starší model a v čase prípravy nášho prostredia ho už nebolo možné kúpiť, rozhodli sme sa pre vlastné, alternatívne riešenie.

Zhodnotili sme, že dostupné a vhodné riešenie bude použitie svetelných led pásov a ich ovládanie pomocou dosky Arduino Nano.

3.1 Požiadavky

Výsledný produkt musí byť schopný ovládať dve nezávislé lampy. Svetlá musia byť dostatočne výrazné, aby dokázali okamžite upútať pozornosť. Ovládanie musí byť možné aj priamo z C++ kódu, ktorý ovláda ostatné súčasti prostredia, aby sa čas od poslania inštrukcie do vykonania akcie minimalizoval. Medzi počítačom a doskou musí prebiehať real-time komunikácia.

3.2 Návrh riešenia

Ako zdroje svetla sme použili dva červené led pásy. Na ovládanie, t.j. vypínanie a zapínanie led pásov sme zvolili dosku Arduino Nano. Doska je napájaná z počítača pomocou USB kábla. Zároveň prepojenie cez USB poskytuje komunikačný kanál medzi počítačom a doskou. Ide o sériový port 115200. Komunikácia potom prebieha tak, že program posiela na sériový port inštrukcie, ktoré majú isté významy. Arduino celý čas, kým je zapnuté, očakáva inštrukciu a v okamihu, keď nejakú prijme, vykoná akciu podľa významu inštrukcie.

3.3 Význam inštrukcií

Ako reprezentáciu inštrukcií, ktoré Arduino dostáva sme zvolili jednociferné čísla, každé so špecifickým významom. Ak je prečítaná hodnota:

- 1, tak sa zareaguje zapnutím ľavého svetla
- 2, tak sa zareaguje zapnutím pravého svetla
- 3, tak sa zareaguje vypnutím ľavého svetla
- 4, tak sa zareaguje vypnutím pravého svetla

Potom je možné na Arduino posielať príkazy cez tento sériový port. Takto sa zabezpečí komunikácia medzi komponentom svetiel a samotným experimentálnym prostredím.

3.4 Knižnica na sériovú komunikáciu

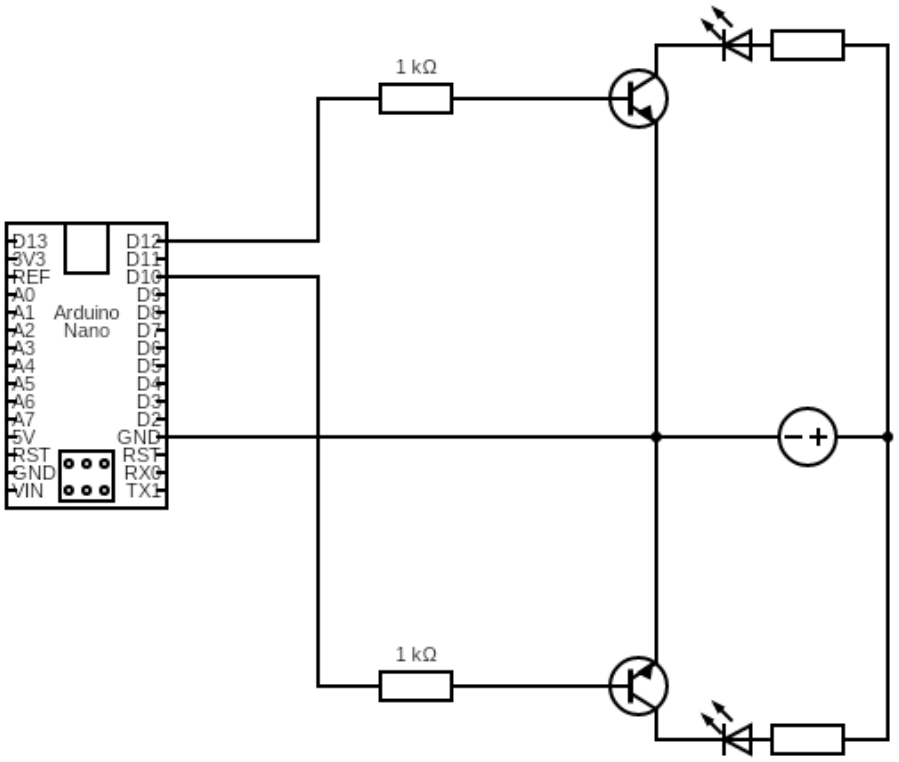
Keďže sme experimentálne prostedie implementovali v jazyku C++, využili sme už existujúcu knižnicu pre sériovú komunikáciu [16]. Knižnica zastrešuje pripojenie sa na sériový port so zadanými parametrami (názov portu, baudRate, ...). Na posielanie hodnôt na sériový port využívame metódu write.

Do knižnice sme doimplementovali ošetrenie výnimky pri zvolení nesprávneho mena portu. Pripadalo nám veľmi zmysluplné, aby bol používateľ softvéru notifikovaný o tomto druhu chyby jasnou a popisnou chybovou hláškou.

3.5 Návrh obvodu

Potrebovali sme vytvoriť dve rovnaké, ale nezávislé lampy. Jednu na reprezentáciu ľavého stimulu, druhú na reprezentáciu pravého. Na napájanie svetelných led pásov bolo potrebné použiť 12V. Arduino Nano je ale schopné napájať len 5V, preto bolo potrebné pridať iný zdroj napájania, a to menič z 220V na 12V. Samotné zapínanie a vypínanie led pásu potom funguje pomocou NPN tranzistora, ktorý je využitý ako spínač obvodu a napája sa 5V z Arduina. Teda keď chceme zapnúť lampu, robíme to tak, že pošleme signál HIGH na príslušný tranzistor, ten zopne obvod s led pásom a ten začne svietiť. Ak chceme lampu vypnúť, pošleme na tranzistor signál LOW a obvod sa rozopne, led pás zhasne.

Podrobná schéma zapojenia je na obrázku 3.1.



Obr. 3.1: Diagram obvodu na ovládanie svetiel

3.6 Realizácia

3.6.1 Program pre Arduino

Arduino sa pripojí na sériový port 115200, toto je možné nastaviť napríklad pomocou prostredia Arduino Ide, popísaného v sekcii 1.2.13.

Tranzistory sú pripojené na digitálne piny 10 a 12, 10 pre ľavé a 12 pre pravé svetlo. Arduino potom v hlavnom loope očakáva inštrukcie prichádzajúce na tento sériový port a na základe inštrukcie posiela príslušný signál HIGH alebo LOW na pin ľavého alebo pravého tranzistora. Zdrojový kód pre Arduino dosku sa nachádza v prílohe A.

3.6.2 Kryty na lampy

Aby neboli lampy pre účastníka rušivé a nesvietili príliš priamo, vyrobili sme plastové polguľové kryty z matného materiálu, do ktorých sme vložili spomínané led pásy. Tieto kryty zabezpečili, že svetlá nesvietia priamo, ale zmena farby je veľmi výrazná a preto aj veľmi rýchlo spozorovateľná.

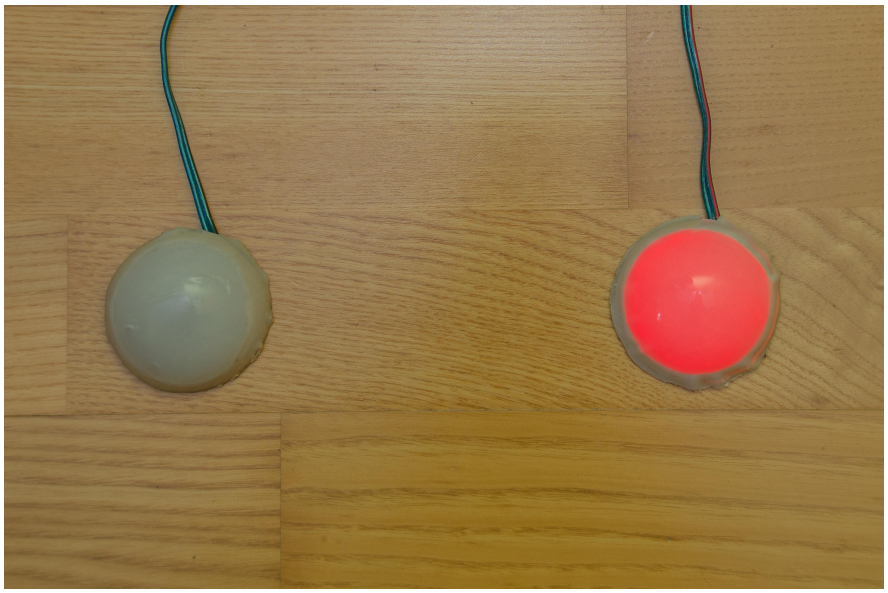
3.6.3 Výsledný produkt

Výsledný produkt bol zostrojený podľa návrhu v tejto kapitole. Produkt sa skladá zo 4 oddelených častí.

- ľavé svetlo s napájacím káblom
- pravé svetlo s napájacím káblom
- arduino s tranzistormi, externým zdrojom napájania a konektormi na lampy
- kábel USB do mini USB (na prepojenie počítača a Arduina)

Pre použitie lámp treba pripojiť ľavé svetlo do konektora s označením 10 a pravé svetlo do konektora s označením 12. Následne zapojiť USB kábel do Arduino dosky a do počítača, z ktorého spúšťame program. Ďalej ešte treba pripojiť externý zdroj napájania lámp do zásuvky.

Výsledný produkt je zobrazený na obrázku 3.2. Ľavé svetlo je na fotografii vypnuté, pravé je zasvietené.



Obr. 3.2: Výsledný produkt - lampy

Kapitola 4

Návrh a realizácia simulovaného prostredia

Prvou súčasťou prostredia na experiment je konzolová aplikácia na prácu s iCubom v Simulátore robota iCub. Robot sa zobrazuje na obrazovke počítača, cez ktorú interaguje s účastníkom experimentu. Súčasťou tohto prostredia sú okrem robota v simulátore aj dve fyzické svetlá, ktoré vytvárajú stimuly a klávesnica, ako nástroj na reakciu účastníka.

4.1 Konkrétne použitie YARP servera

Pre ovládanie simulovaného robota využijeme YARP server. Po nainštalovaní robotology-superbuild máme k dispozícii terminálový príkaz yarpserver, ktorý spustí samotný name server na niektorej lokálnej IP adrese.

Je vhodné sa vopred sa rozhodnúť, na ktorej lokálnej IP a porte by mal bežať, prípadne nechať YARP vybrať si adresu. Ak sme adresu vybrali my, musíme ju nastaviť v konfiguračnom súbore yarp.conf, ktorý sa pravdepodobne nachádza v /.config/yarp/yarp.conf . Súbor obsahuje 1 riadok, s údajmi IP adresa, port a slovo 'yarp' oddelené medzerami, napr. 192.168.0.106 10000 yarp. Ak potom chceme aby server bežal na tejto nami určenej konfigurácii, spustíme yarpserver s prepínačom –read.

Ak nemáme preferenciu na to, kde má server bežať, môžeme spustiť yarpserver s prepínačom –write, pričom si server vyberie niektorú lokálnu IP adresu, zapíše ju do konfiguračného súboru yarp.conf a spustí sa.

Keď je YARP server spustený, môžeme sa napr. pomocou prehliadača pripojiť na danú lokálnu adresu a port a v časti all ports by sme mali nájsť jeden tcp port /root.

Ak server beží, môžeme spustiť simulátor. Opäť to urobíme pomocou konzolového príkazu v samostatnom terminálovom okne. Spustíme príkaz iCub_SIM, čím sa spustí simulátorová aplikácia. V all ports sa objaví približne 50 nových portov s prefixom

/icubSim . Sú to porty ktoré udržiavajú tcp spojenie s jednotlivými ovládačmi kĺbov robota a cez ktoré dokážeme interagovať s prostredím simulátora a kontrolovať jednotlivé kĺby robota v simulátore. Fungovanie môžeme otestovať priamo v prehliadači alebo v programe YARP motor gui.

V prehliadači môžeme otestovať napríklad pohyb hlavy nahor a nadol tak, že nájdeme kĺb /icubSim/head/rpc:i a pošleme mu príkaz 'set pos 0 10' teda ovládame nultú os hlavy, t.j. kĺb nahor a nadol a meníme jeho pozíciu z 0 na 10, bezprostredne po poslaní sa hlava iCuba v simulátore posunie mierne nadol. V programe YARP motor gui môžeme rovnaký kĺb ovládať pomocou slidera v záložke /head, pod názvom JOINT0.

4.2 Ovládanie robota iCub z C++

Na ovládanie robota iCub je potrebné zabezpečiť komunikáciu medzi YARP serverom a C++. Keďže C++ je primárny jazyk na komunikáciu s YARPom, existujú knižnice, ktoré komunikáciu zabezpečujú. Použité boli nasledujúce triedy:

- PolyDriver - A container for a device driver.
- IPositionControl - Interface for a generic control board device implementing position control.
- IEncoders - Control board, encoder interface.

TODO, nejaké vysvetlenie, čo sa tam vlastne deje

4.3 Popis použitých kĺbov

V experimente sme použili pohybovanie hlavy a očí. Ku všetkým týmto kĺbom pristupujeme cez port s prefixom icubSim/head . Na hlave robota iCub je 6 ovládateľných kĺbov, ktoré robia:

1. pohyb hlavy nahor a nadol
2. rotáciu okolo spodku krku, doprava a doľava
3. pohyb hlavy doprava a doľava
4. pohyb očí nahor a nadol
5. pohyb očí doprava a doľava
6. pohyb očí k sebe a od seba

TODO nejaký obrázok trajektórie ktorú oči nasledujú

Na začiatku použijeme kĺby 1 a 4 aby robot posunul hlavu a pohľad trochu nadol. Túto pozíciu nazveme iniciálnou pozíciou.

Pohyb, ktorý robot následne vykonáva je pohnutie hlavou a očami smerom do strany a nadol, pričom upriami pohľad na jedno zo svetiel. Pri tomto pohybe sa používajú kĺby $2, 3, 4$ a 5. Kĺby 2 a 3 sa postupne vychýlia o 10 na príslušnú stranu, tým sa posunie hlava na správne miesto, aby prirodzene pôsobila, že sa pozerá na lampu. Zároveň sa kĺb 4 o 20 nadol a kĺb 5 o 10 do strany, t.j. pohyb očí nadol a do strany.

Na konci daného trialu sa robot vráti do iniciálnej pozície a nasleduje ďalší.

4.4 Overenie skončenia pohybu

Každú požiadavku na pohyb ktorú robotovi pošleme robot vykonáva istý čas. Tento čas potrebuje program počkať. Na zistenie, či sa pohyb už prestal vykonávať použijeme metódu checkMotionDone(bool * flag) triedy IPositionControl, ktorá do flagu vloží hodnotu true ak robot nehýbe žiadnym z jeho kĺbov, false ak niektorým hýbe. Ak by bolo potrebné pýtať sa, či sa pohybuje konkrétny kĺb, existuje rovnomenná metóda, ktorej je možné špecifikovať, na ktorý kĺb sa pýtame.

Čakanie programu potom zabezpečujeme jednoduchým while cyklom, ktorý v každej iterácii overí, či sa robot dohýbal. V okamihu ako zistí, že sa dohýbal, cyklus sa ukončí a môže sa pokračovať ďalšími krokmi.

4.5 Meranie času

Súčasťou experimentu je opakované meranie reakčného času účastníka. Reakčný čas je čas od vytvorenia stimulu, t.j. zasvietenia lampy po zachytenie reakcie, t.j. stlačenie klávesy. Na meranie času sme použili C++ knižnicu chrono a jej objekty system_clock a duration. Čas sme merali jednoducho tak, že si pri vytvorení stimulu zapamätáme aktuálny čas pomocou system_clock metódy now a pri získaní reakcie sme si opäť zapamätali aktuálny čas. Výsledný reakčný čas - inštanciu duration - sme potom získali ako rozdiel koncového času a začiatočného času.

4.6 Odchytávanie vstupu, konzolové výstupy

Aplikácia na vykonávanie experimentu je konzolová. Teda vstupy aj časť výstupov s ktorými pracuje sa posielajú na konzolu terminálu. Na začiatku jedného behu experimentu je potrebné získať od používateľa identifikačné číslo participanta, na základe

ktorého budú identifikované jeho záznamy vo výsledných nameraných dátach. Toto číslo používateľ zadá a potvrdí enterom.

Ďalšie očakávané vstupy budú reakcie participanta, ktoré nebudú potvrdzované enterom, pretože ide o čo najrýchlejšie získanie informácie o reakcii. Reakcia je stlačenie konkrétnej klávesy na klávesnici, teda jeden znak. Aby sme dosiahli, že terminál posiela znaky okamžite po zadaní, nie až po stlačení entera, program pošle terminálu systémovú inštrukciu stty raw.

Okamžite po spracovaní vstupu participanta sa terminál prepne do normálneho režimu pomocou stty cooked a vypíše sa informácia pre používateľa o tom, koľké meranie prebehlo, aká reakcia bola zaznamenaná a aký reakčný čas bol nameraný.

Tesne pred očakávaním ďalšej reakcie sa buffer terminálu aj programu vyprázdni, čím sa vyriešia prípady ak participant pri reakcii zadal viacero znakov, prípadne podržal klávesu a zapísalo sa niekoľko znakov. Následne sa terminál vráti do raw stavu a čaká na reakciu na prichádzajúci stimul.

4.7 CSV výstupy

Zaznamenané reakčné časy sa okrem konzoly musia zapísať aj do súboru. Na ukladanie sme zvolili csv súbory s jednoduchou štruktúrou. Jeden riadok zodpovedá jednému meraniu. Skladá sa z :

- identifikačného čísla účastníka
- poradového čísla trialu
- znaku L alebo R podľa strany na ktorú sa robot otočil (L pre ľavú, R pre pravú)
- znaku L alebo R podľa strany na ktorej zasvietila lampa (L pre ľavú, R pre pravú)
- znaku L alebo R alebo - podľa toho, akú reakciu účastník vykonal (L pre ľavé tlačidlo, R pre pravé tlačidlo, - pre neplatnú reakciu)
- reakčného čas v milisekundách (na najviac 3 desatinné miesta)

Neplatná reakcia znamená, že účastník stlačil iné, než pravé alebo ľavé vyznačené tlačidlo.

Pri každom meraní sa zapisuje do 2 súborov. Do súboru output.csv, ktorý slúži aj ako log toho, v akom poradí išli participanti a do súboru csv súboru, ktorého názvom je identifikačného čísla účastníka. V tomto súbore sa nachádzajú namerané reakcie len pre jedného konkrétneho účastníka s daným identifikačným číslom.

4.8 Spúšťanie programu

Program je potrebné makenúť pomocou CMake. Následne je program spúšťaný z terminálového okna na operačnom systéme Ubuntu 20.04.4 LTS (Focal Fossa). Pri spúšťaní je treba ako parameter zadať meno robota. To je prefix názvov portov pripojených na YARP serveri, napríklad icubSim. Program potom spustíme pomocou ./<nazov maknuteho spustitelneho suboru> –robot <meno robota>, teda napríklad ./simulator –robot icubSim. Ak sa program spúšťa správne, pár sekúnd načítava encodery, o čom nás aj notifikuje a následne sa objaví výzva na zadanie identifikačného čísla účastníka. Po zadaní začne prebiehať experiment s 80 trialmi.

Kapitola 5

Návrh a realizácia prostredia s fyzickým robotom

Táto kapitola úzko súvisí s predchádzajúcou kapitolou o návrhu a prevedení prostredia na interakciu so simulovaným robotom iCub.V tejto kapitole nájdete informácie a postup adaptácie aplikácie pre simulovanú formu stelesnenia na program konzolovej aplikácie pre ovládanie fyzického robota iCub. Vo fyzickej podobe budeme využívať fyzické svetlá [?] a reakcie získavané z klávesnice, podobne ako pri simulovanej forme.

5.1 Vyvíjanie pomocou Gazebo

Pri vyvíjaní robotického softvéru sa pre dostupnosť a bezpečnosť využívajú existujúce softvérové riešenia na simuláciu pohybov robota. Pre vyvíjanie programu sme využili 3D simulátor Gazebo, keďže podľa kolegov, ktorí pracujú s fyzickým iCubom v Prahe sa Gazebo najviac približuje správaniu skutočného robota. Okrem toho, že sme program priebežne testovali v Gazebe, posielali sme ho kolegom na FEL ČVUT v Prahe, kde ho otestovali na skutočnom robotovi a dali nám odporúčania, čo upraviť.

5.2 Gazebo svet

Pre účely testovania bolo vhodné vytvoriť si jeden setup objektov v Gazebo simulátore, aby sme zachovali nemenné podmienky a tak mohli sledovať dosiahnuté zmeny. Do Gazeba sme importovali SDF model robota iCub, konkrétne model iCub Fixed [12]. Po vložení modelu robota sa vytvoria rovnaké porty na YARP serveri ako pri spustení simulátora robota iCub. Pomocou nich potom vieme robota v Gazebe ovládať. Okrem robota sme pridali osvetlenie scény a z Gazeba sme vyexportovali nový SDF model - svet, ktorý dokážeme v Gazebo simulátore kedykoľvek otvoriť a podmienky ostávajú rovnaké.

5.3 Úprava programu pre simulátor

Vďaka ovládaniu kĺbov fyzického robota pomocou platformy YARP bolo možné hlavnú časť funkcionality vytvorenú na ovládanie rotota v simulátore znovupoužiť. V tejto inštancii programu bolo potrebné upraviť a špecifikovať niektoré konštanty (napr. speed, acceleration, delays) tak, aby robot vykonával rovnako vyzerajúce a rovnako dlho trvajúce pohyby v oboch formách.

5.4 Problém s YARP funkciou CheckMotionDone

Pri práci s Gazebom, ako aj fyzickým robotom sme narazili na neočakávané správanie YARP funkcie checkMotionDone(IPositionControl * pos), ktorá by mala overovať, či sa robotov pohyb skončil. Funkciu využívame na overenie skončenia pohybu zadaného kĺbu. Robíme to tak, že sa funkcia volá v nekonečnom cykle, kým nie je jej návratová hodnota rovná true, potom sa cyklus skončí. Toto v simulátore bezproblémovo fungovalo, ale pri spustení v Gazebo alebo na fyzickom robotovi boli niektoré konfigurácie odignorované a pohyb sa nevykonal.

Problém, ktorý sme odhalili bol, že pravdepodobne pri týchto dvoch formách trvá notifikovanie robota o požiadavke na vykonanie pohybu a následné začatie pohybu dlhšie. Teda môže nastať situácia, že sa cyklus spustí ešte predtým, než sa robot začne hýbať, čo ale spôsobí, že sa vyhodnotí, že pohyb už skončil, lebo sa nehýbe.

Riešenie pre túto formu bolo pred kontrolou ukončenia pohybu kontrolovať podobným spôsobom aj začatie pohybu.

5.5 Ovládanie fyzického robota

Fyzický iCub nie je priamo prepojený s počítačom z ktorého spúšťame program. Je pripojený na lokálnu sieť, na ktorej beží YARP server administrovaný z hlavného počítača v laboratóriu. Pre prácu s robotom sa stačí pripojiť na lokálnu sieť, napríklad pomocou sieťového kábla a spustiť príkaz yarp detect, ktorý detekuje všetky dostupné bežiace YARP servery. Po nájdení spusteného servera sa nám zobrazí informácia o tom, na akej adrese a porte beží a opäť je možné skontrolovať, ktoré porty sú pripojené na tomto YARP serveri. Potom už nie je nutné spúšťať vlastný, lokálny server, ale po detekcii a nájdení servera na lokálnej sieti, budú YARP knižnice pre C++ automaticky pracovať a nájdeným YARP name serverom.

Kapitola 6

Návrh virtuálneho prostredia

Prepojenie robota iCub s virtuálnou realitou bol úplne nový a nepreskúmaný problém. Aktuálne možnosti práce s virtuálnou realitou nie sú veľmi prispôbené na využitie vo výskume. Cieľovou skupinou výrobcov náhlavných displejov sú skôr hráči počítačových hier.

V nasledujúcej kapitole sme preskúmali niektoré existujúce riešenia pre prepojenie robota s virtuálnou realitou. Následne v kapitole nájdete návrh vhodnej technológie na prepojenie robota iCub s virtuálnym náhlavným displejom HTC Vive PRO, ako aj technické detajly návrhu virtuálneho prostredia s robotom, stolom a ovládateľnými lampami.

6.1 Existujúce spôsoby prepojenia VR a robota

Našli sme a porovnali 3 exitujúce riešenia na prepojenie virtuálnej reality a robotického softvéru. Ide o projekt Unity3D Robotics Toolkit z prostredia Univerzity v Hamburgu a 2 na seba nadväzujúce projekty, VRUI MDF a OpenVR headset ROS, ktoré vznikli na Univerzite v Cincinnati.

6.1.1 Unity3D Robotics Toolkit

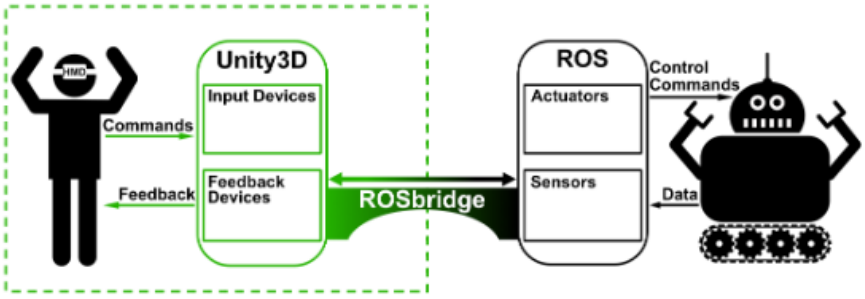
[17] Prvým skúmaným riešením je Unity3D Robotics Toolkit, open-source nástroj prepájajúci Robot Operating System (ROS) s prostedím Unity3D. Prepojenie je riešené pomocou existujúceho ROSbridge, ktorý je obsiahnutý v ROS package. Tento projekt implementuje ROSbridge interface v Unity3D na strane človeka a prepája ho s ROSbridge node na strane ROS a teda robota.

Unity3D od verzie 2017.3 podporuje prepojenie s HTC Vive Pro headsetom aj pod operačným systémom Linux.

ROS využíva na popis robota Unified Robot Description Format (URDF). Ide o XML textový formát, ktorý definuje vizuálne aj fyzikálne vlastnosti. URDF modely

robota iCub sú dostupné na githube [13].

todo icub sa nekamarati s rosom, radsej yaprs, icub mal nejaky problem s unity, treba najst a zdvovodnit, ale povodna issue zmizla, resp neexistuje povodny link, skusila som aj web archiv a nic

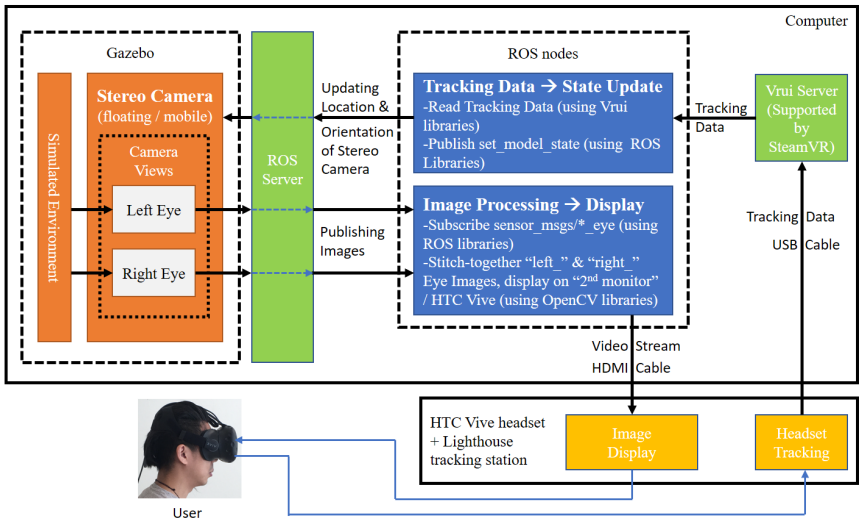


Obr. 6.1: Architektúra softvéru Unity3D Robotics Toolkit

6.1.2 VRUI MDF

[18] Ďalší softvér, ktorý sme skúmali je VRUI MDF. Tento prepája HTC Vive so simulátorom Gazebo, ktorý je veľmi vhodný pre prácu s iCubom. Na prepojenie používa VRui server, ROS, SteamVR for Linux a OpenCV.

Projekt bol určený na účel výuky na univerzite. Autori Shi a McGhan[18] uvádzajú, že tento setup funguje dobre pre niektoré edukačné projekty a pre menej komplexné robotické modely. Tento softvér má ale problém so simuláciou komplexnejších robotov, čo sa prejavuje spomalením simulácie. Preto úplne nevyhovuje na testovanie humanoidných robotov. Autori uvádzajú aj analýzu spomalenia. [18]



Obr. 6.2: Architektúra softvéru VRUI MDF

6.1.3 Open VR headset ROS

Posledným skúmaným riešením je projekt OpenVR headset ROS, ktorý stavia na softvéri VRUI MDF. Architektúra ostáva podobná ako v diagrame na obrázku 6.2, s tým rozdielom, že komponent VRui server bol vymenený za softvér OpenVR. OpenVR zaznamenáva informácie o zmene polohy virtuálneho zariadenia a posiela obraz do virtuálneho headsetu. Ide o stále živý projekt, ktorý sa vyvíja. Aktuálna verzia a informácie k používaniu sú popísané v githubovom repozitári [19].

Open VR headset ROS bol otestovaný aj s virtuálnym headsetom HTC Vive Pro, jeho ovládačmi a base stanicami Lighthouse 1.0 aj 2.0. , ktorý využívame.

Rozhodli sme sa, že toto riešenie je pre náš headset a robota iCub najvhodnejšie a preskúmame ho hlbšie.

6.2 Opis stroja na prácu s virtuálnou realitou

Na ovládanie VR bol potrebný výkonný počítač, ktorý zvládne sledovanie polohy headsetu a ovládačov, ako aj renderovanie obrazu zobrazeného v náhlavnom displeji. Použili sme laptop ACER Predator Helios 300 s procesorom Intel Core i7-10750H, grafikou NVIDIA GeForce RTX 2070 with Max-Q Design a 32 GB RAM. Pracovali sme pod operačným systémom Ubuntu 20.04.4 LTS (Focal Fossa).

6.3 Prerekvizity softvéru

Softvér OpenVR headset ROS prepája niekoľko existujúcich softvérov. Na komunikáciu s virtuálnou realitou používa SteamVR a OpenVR. Na zobrazovanie prostredia a robota používa simulátor Gazebo a na prepojenie ROS node.

6.4 SteamVR

Na prácu s virtuálnou realitou sme použili softvérový nástroj SteamVR, ktorý podporuje viaceré virtuálne headsety, vrátane nami používaného HTC Vive Pro.

SteamVR je jednoducho a bezplatne dostupný cez platformu Steam. SteamVR sa snaží byť jednoduchý na ovládanie tak, aby s ním mohol pracovať aj bežný používateľ, preto by po nainštalovaní malo stačiť pripojiť headset a spustiť aplikáciu a všetko by malo fungovať.

Keďže sme ale pracovali pod Linuxom, ktorý nie je stopercente podporovaný, pri tomto procese nastalo niekoľko komplikácií.

6.4.1 Komplikácie so SteamVR

Linux je podporovaný iba vo verzii SteamVR Beta, preto je pre používanie VR potrebné prepnúť sa do tejto verzie. Používanie nesprávnej verzie spôsobuje chybu 'headset not detected', z ktorej ale nie je úplne jasné, kde je problém.

Ďalším problémom bolo, že aj keď bol headset už detekovaný, v náhlavnom displeji sa nezobrazovalo nič. Na displeji počítača však bolo možné vidieť obraz, ktorý mal byť premietaný v headsete. Bolo zrejmé, že počítač s headsetom komunikuje, keďže obraz na displeji počítača sa menil v závislosti od polohy headsetu. Na základe toho sme objavili chybu v samotnom GUI aplikácie SteamVR, ktorá oznamovala, že sa treba prepnúť do 'Direct display mode', ale pri stlačení tlačidla na prepnutie softvér neurobil nič.

Na Steam fórach sme sa dozvedeli, že ide o známy nedostatok v SteamVR Beta pre Linux, pričom tieto funkcionality nie sú funkčné, napriek tomu, že sú pre ne tlačidlá v grafickom rozhraní. Ďalšou informáciou bolo, že prepnutie do 'Direct display mode' by sa malo udiať automaticky, ak je možné. Dôvod, prečo automatické prepnutie nebolo možné bol, že verzia grafického driveru ktorý sme používali bola príliš zastaralá. Riešením bolo prejsť na verziu nvidia-510. Potom už bol displej schopný prejsť do 'Direct display mode' a zobrazovať obraz správne.

6.5 Testovanie softvéru Open VR headset ROS

Keďže išlo o novovyvinutý a málo testovaný softvér, po stiahnutí z repozitáru ho nebolo možné maknúť. Zdalo sa, že mal problém s linkovaním ciest na jednotlivé súčasti softvéru, ale keďže išlo o slabo popísaný a pomerne komplexný softvér, nedokázali sme problém opraviť.

Kontaktovali sme preto autorov softvéru, ktorí sa pokúsili o vytvorenie novej, stabilnejšej verzie a spoločne sme ju testovali. Podarilo sa nám nájsť a odstrániť niekoľko nedostatkov a následne vytvoriť a otestovať verziu, ktorá je aktuálne dostupná na github projektu [19] v branchi devel-cm.

Táto nová verzia spoľahlivo sledovala zmenu polohy headsetu a haptických ovládačov, ale renderovanie obrazu v náhlavnom displeji nebolo funkčné a displej zobrazoval error. Error sa prejavoval tak, že displej zobrazoval len žltú farbu s červeným nápisom Message log.

Od autorov sme dostali informáciu, že podobný problém sa im podarilo vyriešiť zmenou rozlíšenia v akom softvér renderuje obraz pre náhlavný displej, no na našom stroji táto zmena nepomohla vyriešiť danú chybu.

Na základe našej spolupráce s autormi sme im dali množstvo podnetov, na ktorých pracujú, a ktoré dopomôžu budúcemu k fungovaniu softvéru.

6.6 Svetlá vo VR

Od návrhu fyzických svetiel opísanom v časti 3.2 sa odvíjal aj návrh svetiel zobrazovaných vo virtuálnom prostredí, aby ostali podmienky vo všetkých stelesneniach čo najpodobnejšie. Virtuálne prostredie je vytvárané v 3D simulátore Gazebo. Svetlá majú za účel vytvoriť stimul zmenou farby.

V simulátore Gazebo existujú svetlá, ktoré sú určené na osvetlenie scény, tie sú ale statické a tento model nepodporuje vypínanie a zapínanie týchto svetiel.

Pre Gazebo existuje Flashlight Plugin, ktorý umožňuje umiestniť do scény zdroj svetla. Plugin umožňuje nastaviť fixné intervaly zapínania a vypínania aj farbu v daných intervaloch. Keďže sú tieto intervaly fixné, ale naše prostredie musí dokázať reagovať dynamicky, vypínanie a zapínanie v pevne stanovených intervaloch nebude možné pre naše účely použiť.

todo riesenie je spravit polgule a menit ich farbu, pripadne spravit biele polgule a cervenu polgulu a menit ich polohu, pripadne spravit biele polgule a cervene svetlo a menit polohu svetla

6.7 Gazebo svet pre VR

Gazebo svet pre virtuálne prostredie vychádza z testovacieho Gazebo sveta Gazebo svet. Do sveta sme vložili stôl [20], na pozíciu, na ktorej sa nachádzal v reálnej miestnosti reálny stôl. **todo, co so svetlami**

Kapitola 7

Zber a analýza dát

V tejto kapitole uvádzame praktické využitie navrhnutého a vyvinutého riešenia v meraní reakčného času v psychoogickom experimente. Dáta pre tento experiment boli zbierané na Fakulte matamatiky, fyziky a informatiky Univerzity Komenského v Bratislave a na Fakulte elektrotechniky ČVUT v Prahe. Následne uvádzame výsledky analýzy zozbieraných dát.

7.1 Setup experimentu

Účastník experimentu sedel na jednej strane stola. Na opačnej strane sa nachádzal robot otočený čelom k účastníkovi. Na stole boli umiestnené dve svetlá vo vzdialenosti 50 cm od robota. Okrem svetiel bola na stole položená klávesnica Logitech K120 vo vzdialenosti 50 cm od svetiel, smerom k účastníkovi. Svetlá sme umiestnili napravo a naľavo na základe obrazu z kamery v očiach fyzického robota tak, aby pri úplnom vychýlení do strany bolo príslušné svetlo v strede obrazu z kamery. Teda aby sa naň robot pozeral čo najpresnejšie a najpresvedčivejšie. Na opačnej strane stola bol umiestnený robot otočený ku stolu, vrch hlavy robota bol zarovnaný na vrch hlavy človeka pri sedení na stoličke na opačnom konci stola. Vo fyzickom stelesnení išlo o fyzického robota, v simulovanom išlo o robota na 65 palcovej LED obrazovke Samsung UE65TU7172.

Aby sme zachovali čo najrovnakejšie podmienky pri oboch formách stelesnenia, prispôsobili sme veľkosť robota na obrazovke veľkosti skutočného robota. V oboch formách bolo temeno hlavy robota vo výške 135 cm a z pozície účastníka bolo vidieť hlavu, hruď a hornú časť nôh robota iCub.

7.2 Priebeh merania

Účastník po príchode vyplnil informovaný súhlas s meraním a spracovaním dát - reakčného času. Zároveň si prečítal textovú podobu opisu priebehu experimentu a inštrukcie.

Účastník bol informovaný o jeho úlohe, čo najrýchlejšie a najpresnejšie reagovať na zasvietenie lampy stlačením príslušného tlačidla. Potom sa účastník presunul na miesto merania, kde si sadol za stôl za ktorým sa na opačnom konci nachádzal robot. Tu mu bolo slovne zopakované ako experiment prebieha a zodpovedané prípadné nejasnosti zo strany účastníka. Následne prebehlo meranie všetkých 80 trialov. Po skončení merania každý účastník vyplnil dotazník, ktorý sa týkal vnímania experimentu a výzoru, pôsobenia robota v experimente.

7.3 Zber dát

Pilotné meranie simulovanej formy prebehlo na FMFI UK v Bratislave, v marci 2022. Namerali sme 15 osôb. Pri meraní sme použili fyzické svetlá a klávesnicu, pričom na zobrazenie simulovaného robota sme použili menší 20.1 palcový LCD monitor SAMSUNG Sync Master 205BW . Dáta sme nakoniec pri vyhodnocovaní nepoužili, pretože sme chceli aby podmienky (prostredie, okolie, svetelné podmienky, ...) pre meranie boli čo najrovnakejšie. Meranie sme preto zopakovali, a to v rovnakom prostredí ako meranie s fyzickou formou.

Hlavné meranie sa uskutočnilo v apríli 2022 na FEL ČVUT v Prahe. Tu sa počas 3 dní namerali dáta pre simulovanú aj fyzickú formu.

Obr. 7.1: Setup fyzického (naľavo) a simulovaného (napravo) prostredia pre experiment

7.4 Opis nameranej vzorky

Merania sa zúčastnilo 45 osôb, z toho 22 pre simulovanú formu a 23 pre fyzickú formu. Z výslednej vzorky simulovanej formy musela byť 1 osoba odstránená z dôvodu prerušenia experimentu. Z fyzickej vzorky museli byť odstránené 2 osoby, z dôvodu poruchy kĺbu fyzického robota, ktorá spôsobila odlišnosť od ostatných behov experimentu.

Výsledná vzorka teda obsahova 21 osôb pre simulovanú a 21 pre fyzickú formu.

todo zeny, muzi, vek ...

7.5 Analýza

Táto práca je súčasťou rozsiahlejšieho projektu, ktorý zahŕňa aj diplomovú prácu študentky kognitívnej vedy na Fakulte matamatiky, fyziky a informatiky Univerzity Komenského, Kassandry Friebe. Práve v jej diplomovej práci sa nachádza rozsiahla analýza nameraných dát.

todo strucne info o analyze + pekne grafy ?

7.6 Výsledky

On average, participants in the copresent robot condition responded faster on congruent trials (M = 291 ms) than on incongruent trials (M = 305 ms).

todo

Záver

Na záver už len odporúčania k samotnej kapitole Záver v bakalárskej práci podľa smernice „V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.“

Všimnite si správne písanie slovenských úvodzoviek okolo predchádzajúceho citátu, ktoré sme dosiahli príkazmi `\glqq` a `\grqq`.

V informatických prácach niekedy býva záver kratší ako dve strany, ale stále by to mal byť rozumne dlhý text, v rozsahu aspoň jednej strany. Okrem dosiahnutých cieľov sa zvyknú rozoberať aj otvorené problémy a námety na ďalšiu prácu v oblasti.

Abstrakt, úvod a záver práce obsahujú podobné informácie. Abstrakt je kratší text, ktorý má pomôcť čitateľovi sa rozhodnúť, či vôbec prácu chce čítať. Úvod má umožniť zorientovať sa v práci skôr než ju začne čítať a záver sumarizuje najdôležitejšie veci po tom, ako prácu prečítal, môže sa teda viac zamerať na detaily a využívať pojmy zavedené v práci.

TODO

využitie softvéru, prinosy prace, Future work, nico, ruky, eye tracker

Bibliografia

[1] Christoph Bartneck et al. *Human-Robot Interaction: An Introduction*. Cambridge University Press, feb. 2020. ISBN: 9781108735407. DOI: 10.1017/9781108676649.

[2] S Šabanović. „Robots in Society, Society in Robots“. In: zv. 2. Okt. 2010, 439–450. ISBN: 0-7803-9505-0. DOI: 10.1007/s12369-010-0066-7.

[3] Jamy Li. „The benefit of being physically present: A survey of experimental works comparing copresent Robots, telepresent Robots and virtual Agents“. In: *International Journal of Human-Computer Studies* 77 (jan. 2015). DOI: 10.1016/j.ijhcs.2015.01.001.

[4] Eva Wiese, Patrick P. Weis a Daniel M. Lofaro. „Embodied social robots trigger gaze following in real-time HRI“. In: *2018 15th International Conference on Ubiquitous Robots (UR)* (2018), s. 477–482. DOI: 10.1109/URAI.2018.8441825.

[5] Henny Admoni a Brian Scassellati. „Social Eye Gaze in Human-Robot Interaction: A Review“. In: *J. Hum.-Robot Interact.* 6.1 (2017), 25–63. DOI: 10.5898/JHRI.6.1.Admoni.

[6] *Oficiálna webstránka robota iCub*. [Citované 2022-9-1] Dostupné z https://www.iit.it/web/icub.

[7] *Dokumenácia k robotovi iCub*. [Citované 2022-9-1] Dostupné z https://icub-tech-iit.github.io/documentation/.

[8] Ricardo Beira et al. „Design of the robot-cub (iCub) head“. In: zv. 2006. Feb. 2006, s. 94 –100. ISBN: 0-7803-9505-0. DOI: 10.1109/ROBOT.2006.1641167.

[9] V. Tikhanoff et al. „An Open-Source Simulator for Cognitive Robotics Research: The Prototype of the ICub Humanoid Robot Simulator“. In: *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*. PerMIS '08. Gaithersburg, Maryland: Association for Computing Machinery, 2008, 57–61. ISBN: 9781605582931. DOI: 10.1145/1774674.1774684. URL: https://doi.org/10.1145/1774674.1774684.

[10] Giorgio Metta et al. „The iCub humanoid robot: An open platform for research in embodied cognition“. In: *Performance Metrics for Intelligent Systems (PerMIS) Workshop* (jan. 2008). DOI: 10.1145/1774674.1774683.

[11] *Oficiálna webstránka Gazebo simulátora.* [Citované 2022-9-1] Dostupné z `https://gazebosim.org/`.

[12] *Modely robota iCub.* [Citované 2022-9-1] Dostupné z `https://github.com/robotology/icub-gazebo`.

[13] *Nové modely robota iCub.* [Citované 2022-9-1] Dostupné z `https://github.com/robotology/icub-models`.

[14] *Robotology superbuild.* [Citované 2022-9-1] Dostupné z `https://github.com/robotology/robotology-superbuild`.

[15] Bc. Martin Varga. „Emulácia kože v simulátore humanoidného robota iCub a autonómna explorácia tela“. [Citované 2022-9-1] Dostupné z `http://cogsci.fmph.uniba.sk/~farkas/theses/martin.varga.dip16.pdf`. Dipl. pr. Fakulta matematiky, fyziky a informatiky Univerzity Komenského v Bratislave, 2016.

[16] *Knižnica Serialport.* [Citované 2022-9-1] Dostupné z `https://gist.github.com/nuft/d625fae77d43974a48e07c39441b370b`.

[17] Dennis Krupke et al. „Prototyping of Immersive HRI Scenarios“. In: okt. 2017, s. 537–544. DOI: `10.1142/9789813231047_0065`.

[18] Zhenyu Shi a Catharine L. R. McGhan. „Affordable Virtual Reality Setup for Educational Aerospace Robotics Simulation and Testing“. In: *Journal of Aerospace Information Systems* 17.1 (2020), s. 66–69. DOI: `10.2514/1.I010723`. eprint: `https://doi.org/10.2514/1.I010723`. URL: `https://doi.org/10.2514/1.I010723`.

[19] *Projekt OpenVR headset ROS.* [Citované 2022-9-1] Dostupné z `https://github.com/ConradKent/openvr_headset_ros`.

[20] *SDF model stola.* [Citované 2022-9-1] Dostupné z `https://gist.github.com/mkoval/31ec1e75bb09aceb3f79`.

Príloha A: Program pre Arduino

todo popis

```
int x;
void setup() {
 Serial.begin(115200);
 Serial.setTimeout(1);
 pinMode(10, OUTPUT);
 pinMode(12, OUTPUT);

}
void loop() {
 while (!Serial.available());
 x = Serial.readString().toInt();
 if(x == 1)
 {
  digitalWrite(10, HIGH);
 }
 if(x == 2)
 {
  digitalWrite(12, HIGH);
 }
 if(x == 3)
 {
  digitalWrite(10, LOW);
 }
 if(x == 4)
 {
  digitalWrite(12, LOW);
 }
```

}