CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

VR-styrsystem för humanoid robot

En Wizard of Oz-lösning för den sociala roboten Pepper

Kandidatarbete inom Data- och informationsteknik

Timmy Bolinder
Melker Forslund
Elias Lycke
Tobias Pettersson
Marcus Saméus
Robin Åsberg

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Kandidatarbete 2025

VR-styrsystem för humanoid robot

En Wizard of Oz-lösning för den sociala roboten Pepper

Timmy Bolinder
Melker Forslund
Elias Lycke
Tobias Pettersson
Marcus Saméus
Robin Åsberg

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Data- och informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
Göteborg, Sverige 2025

VR-styrsystem för humanoid robot
En Wizard of Oz-lösning för den sociala roboten Pepper
Timmy Bolinder, Melker Forslund, Elias Lycke,
Tobias Pettersson, Marcus Sameus, Robin Åsberg

Handledare: Sofia Thunberg, Institutionen för Data- och informationsteknik
Examinatorer: Patrik Jansson och Arne Linde, Institutionen för Data- och informationsteknik
Rättande lärare: Jasmina Maric, Institutionen för Data- och informationsteknik

Kandidatarbete 2025
Institutionen för Data- och informationsteknik
Chalmers tekniska högskola och Göteborgs universitet
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Bild på Pepper, den sociala roboten som användes i detta projekt.

Typsättning i LaTeX
Göteborg, Sverige 2025

VR-styrsystem för humanoid robot
En Wizard of Oz-lösning för den sociala roboten Pepper
Timmy Bolinder, Melker Forslund, Elias Lycke,
Tobias Pettersson, Marcus Sameus, Robin Åsberg

Institutionen för Data- och informationsteknik
Chalmers tekniska högskola och Göteborgs universitet

Sammandrag

Detta arbete presenterar utvecklingen och utvärderingen av ett styrsystem som använder VR-utrustning för att kontrollera den humanoida roboten Pepper. Styrsystemet är avsett för framtida studier inom människa-robotinteraktion och bygger på Wizard of Oz-metoden, där illusionen av robotens autonomi skapas genom att en mänsklig operatör styr den i hemlighet. Systemet möjliggör teleoperation genom att spegla operatörens förflyttelse inom rummet, vridning av kropp och huvud, samt armrörelser direkt till roboten.

Arbetet utvärderades genom en användarstudie uppdelad i två grupper, en med fokus på användare som fick interagera med roboten, och en där operatörer fick testa styrsystemet. Resultaten visade att ett VR-baserat styrsystem kunde implementeras och spegla operatörens rörelser till roboten. Dock identifierades flera förbättringsområden vad gäller användarvänlighet och inlevelse. Begränsningar orsakades bland annat av systemlatens, begränsad visuell återkoppling samt sensorer från hårdvaran. Användare som interagerade med roboten uppfattade den bland annat som maskinlik, intelligent och autonom.

Abstract

This work presents the development and evaluation of a control system that uses VR equipment to operate the humanoid robot Pepper. The control system is intended for future studies in human-robot interaction and is based on the Wizard of Oz method, where the illusion of robot autonomy is created by having a human operator control the robot covertly. The system enables teleoperation by mirroring the operator's movements within the room, body and head rotation, as well as arm movements directly onto the robot.

The system was evaluated through a user study divided into two groups, one focusing on users interacting with the robot, and one where operators tested the control system. The results showed that a VR based control system could be implemented and mirror the operator's movements onto the robot. However, several areas for improvement were identified in terms of usability and immersion. Limitations were caused by factors such as system latency, limited visual feedback, and sensor constraints in the hardware. Users who interacted with the robot perceived it, among other things, as machine-like, intelligent, and autonomous.

Nyckelord: Pepper, Social robot, Wizard of Oz, WoZ, VR, AR, NAOqi, Unity, HRI

Författarnas tack

Först och främst vill vi rikta ett stort tack till vår handledare Sofia Thunberg, samt alla hjälpsamma personer på avdelningen för Interaktionsdesign och Software Engineering vid Chalmers tekniska högskola. Vi vill även tacka alla deltagare som tog sig tid att medverka i studien, deras insats bidrog inte bara till resultatet, utan gav oss även en djupare förståelse inom området.

- Timmy Bolinder, Melker Forslund, Elias Lycke, Tobias Pettersson, Marcus Saméus, Robin Åsberg, Göteborg, juni 2025

Förkortningar

AI *Artificiell Intelligens*
Artificiell intelligens är datorers förmåga att efterlikna mänsklig intelligens.

API *Application Programming Interface*
Ett applikationsprogrammeringsgränssnitt som innehåller funktioner och kommandon för programmering.

FK *Framåtriktad kinematik*
Framåtriktad kinematik är en beräkningsmetod inom robotik som används för att beräkna en ändeffektors position utifrån givna ledvinklar.

FPS *Frames Per Second*
Bilder per sekund, även kallat bildfrekvens.

IK *Invers kinematik*
Invers kinematik är en beräkningsmetod inom robotik som används för att räkna ut vilka ledvinklar som sätter en robots arm eller ledade system till en viss position.

MRI *Människa-Robotinteraktion*
Människa-Robotinteraktion, eller på engelska Human-Robot Interaction (HRI), är studien av hur människor och robotar kommunicerar och samarbetar med varandra.

NARS *Negative Attitude towards Robot Scale*
NARS är ett psykologiskt mätverktyg som används för att bedöma människors negativa attityder gentemot robotar, särskilt inom områden som människa-robotinteraktion.

P2P *Peer-to-Peer*
Peer-to-Peer är en nätverksmetod där noder kommunicerar direkt med varandra utan en central server.

RGB *Red Green Blue*
RGB är en färgmodell som bygger på additiv färgblandning av rött, grönt och blått ljus. Det används ofta i digitala bildskärmar och pixelformat för att representera färger.

SDK *Software Development Kit*
Innehåller en samling verktyg och bibliotek som API:er, kompilatorer och emulatorer. Används för att utveckla programvara för specifika plattformar eller ramverk.

TCP *Transmission Control Protocol*
Ett nätverksprotokoll som garanterar att all data når mottagaren i rätt form och ordning.

UDP *User Datagram Protocol*
Ett nätverksprotokoll för snabb dataöverföring, men som inte garanterar att data når mottagaren.

VR *Virtual Reality*
Virtual Reality skapar en interaktiv, datorgenererad miljö som upplevs som verklig med hjälp av sensorer, VR-headset och handkontroller.

WoZ *Wizard of Oz*
Begrepp för fjärrstyrd användning av ett till synes autonomt system.

Innehåll

Figurer

Tabeller

1

Inledning

Framsteg inom AI, maskininlärning och robotik har bidragit till en växande utveckling av humanoida robotar [1], där användningsområden inom vård, service och utbildning blivit allt mer aktuella de senaste åren [2], [3]. Bara inom området servicerobotar ses en global marknadstillväxt från 36.2 miljarder dollar år 2022 till 103.3 miljarder dollar år 2026 [4]. I takt med denna utveckling har vikten av forskning på människa-robotinteraktioner (MRI) ökat där interaktionen mellan människa och autonoma humanoida robotar är ett relevant segment [5], [6], [7]. Att utföra MRI-studier på autonoma robotar har dock vissa komplikationer då det fortfarande behövs framsteg i bland annat artificiell intelligens [8]. Istället används ofta Wizard of Oz (WoZ) som en metod för att simulera interaktionen [9], [10]. Den går ut på att en människa tror att hen interagerar med en autonom maskin, men att det egentligen är en operatör som styr. Genom denna metod kan forskare analysera hur människor agerar runt autonoma robotar, även om utvecklingen eller resurser för dessa ännu inte är här idag. Med detta kandidatprojekt ämnar gruppen utveckla och undersöka ett nytt styrsystem som ska användas för WoZ.

I ett tidigare arbete av Thellman m.fl. [11] skapades ett styrsystem för WoZ med antagandet att "den bästa metoden för att simulera autonomi i social människa-robotinteraktion är att, så nära som möjligt, låta en mänsklig operatör övervaka interaktionen och styra roboten som om han eller hon själv befann sig i robotens position". Styrsystemet utvecklades för den humanoida roboten Pepper [12], där en operatör utrustad med VR-headset och handkontroller kunde spegla sina egna rörelser på roboten i realtid. Genom visuell återkoppling från robotens kamera till operatörens utrustning möjliggjordes teleoperation användandes virtual reality (VR) [13], vilket tillät operatören att interagera med omgivningen som om robotens kropp vore deras egna. Styrsystemet hade dock ett flertal begränsningar. Gångrörelserna var baserade på en pekplatta på operatörens handkontroll, videoflödet kom från en extern kamera som saknade djupseende och ingen funktion för att operatören skulle höra en interagerande användare via mikrofon var implementerad. Dessa delar är en direkt begränsning för operatörens upplevelse av systemet, vilket i sin tur också kan ha en påverkan på användarens uppfattning om robotens autonomi. Ett beaktande av dessa begränsningar har gjorts i detta kandidatarbete, där motivationen varit att skapa ett förbättrat styrsystem.

1.1 Syfte och frågeställningar

Syftet med detta arbete är att implementera en VR-lösning för att med helkroppsrörelser kunna styra den sociala roboten Pepper. Roboten ska kunna följa operatörens rörelser när det gäller både rörelse i rummet samt armar och huvud. Efter den praktiska implementeringen av systemet ska en användarstudie genomföras för att utvärdera både operatörens upplevelse av interaktionen och användarupplevelsen hos de personer som interagerar med roboten. Denna implementation och användarstudie görs som en grund för att senare kunna besvara följande frågeställningar:

1. Hur kan man implementera ett VR-baserat styrsystem för Pepper?
2. Hur upplever operatörer styrsystemet?
3. Hur upplevs interaktionen med roboten?

Efter att detta projekt är slut skall MRI-forskare på Chalmers kunna använda denna implementation och resultat som en del av sina studier. För att framtida användare ska kunna förstå alla ingående delar i systemet kommer en manual att skrivas.

1.2 Avgränsningar

Följande avgränsningar har gjorts utifrån den givna tidsramen för kandidatarbetet i kombination med den fysiska utrustningens begränsningar:

- Pepper roterar kroppen endast när operatören vridit sitt huvud 45°horisontellt åt godtyckligt håll.
- Operatören kan ta emot ljud från Pepper, men ej vice versa.
- Inga implementationer kommer göras för att hindra Peppers armars kontakt med fysiska objekt.
- För videoflödet sker ett utbyte av högre bildfrekvens på bekostnad av lägre upplösning.
- Begränsat antal sensorer leder till uppskattning av positionering för vissa kroppsdelar baserat på beräknade estimeringar.
- Endast Python 2.7 32-bit kan användas.

2
Teori

Detta kapitel syftar till att beskriva den teoretiska bakgrund som krävs för att förstå det arbete som genomförts i projektet. Kapitlet behandlar bland annat nätverk, WoZ, MRI och matematiska modeller som framåtriktad samt invers kinematik.

2.1 Människa-robotinteraktion

Inom området MRI har Thomas B. Sheridan [14] identifierat fyra huvudsakliga tillämpningsområden. Det första är mänsklig övervakning av robotar som utför rutinuppgifter, där robotar arbetar självständigt med repetitiva och förutsägbara uppgifter, medan en människa övervakar och ingriper vid behov. Det andra är fjärrstyrning av fordon vid orutinerade uppgifter i farlig eller svårframkomlig miljö. Det tredje gäller automatiserade fordon med mänskliga passagerare, som självkörande bilar och andra autonoma transportsystem. Det fjärde och sista området är social interaktion mellan människa och robot, och handlar om robotar utformade för att interagera med människor på ett naturligt och intuitivt sätt i sociala sammanhang, till exempel som assistenter i hemmet, inom vården eller utbildningssektorn.

Det sistnämnda området syftar till att hjälpa robotutvecklare att skapa styrsystem vars beteendemönster upplevs som mänskliga att interagera med. Sådana robotar ska inte enbart kunna följa instruktioner, för att upplevas som mänskliga hjälper det ifall de kan tolka eller använda sig av kroppsspråk, tonläge, och ögonkontakt [15], [16], [17]. Detta kräver avancerad teknik inom artificiell intelligens, maskininlärning och sensorer som gör det möjligt att uppfatta och analysera mänskligt beteende i realtid. Robotars förmåga att tolka mänskliga signaler är en avgörande faktor för att de ska uppfattas som empatiska, pålitliga och tillgängliga i sociala sammanhang [18].

Sociala robotar har redan börjat tillämpas inom skolor, restauranger, butiker och vården. Ett exempel på detta är roboten Hobbit [19], som utvecklades för att hjälpa äldre att leva självständigt i så stor mån som möjligt [20]. Hobbit kan plocka upp saker från golvet, hämta föremål från höga hyllor och hjälpa användaren att leta efter förlorade objekt. Den kan även larma hemtjänsten vid fallolyckor och bidra med social stimulans. Ett annat exempel är QTrobot, framtagen för att stödja barn med autism [21], som har visat sig öka elevernas engagemang och minska ångest samt sensorisk överbelastning tack vare sitt förutsägbara och familjära beteende [22].

Även roboten Pepper klassificeras som en social robot [23]. Pepper är utformad för

att samverka med människor i miljöer där social interaktion är central, exempelvis inom vård, utbildning och service [24]. Med hjälp av sensorer, kameror och mikrofoner kan Pepper registrera och tolka ansiktsuttryck, kroppsspråk och tonläge [25], och därefter anpassa sitt beteende för att främja en mer naturlig interaktion. I detta projekt kommer Peppers förprogrammerade sociala funktioner inte att utnyttjas. Robotens rörelser styrs manuellt av en operatör, vilket möjliggör en undersökning av styrsystemets funktionalitet utan påverkan från robotens egna beteenden.

2.2 Wizard of Oz-metoden

I de fall när man inom MRI vill forska på interaktion mellan människa och autonoma robotar är det vanligt att använda metoden WoZ [9]. Vid utförandet av metoden får en användare interagera med en maskin som är teleopererad av en mänsklig operatör, men där användaren ej vet om detta [10]. Syftet är att simulera interaktionen mellan användare och ett system som ofta ännu inte är implementerbart. Kelley [26], som ursprungligen använde WoZ-metoden [9], studerade det mänskliga agerandet på ett datorsystem som var mer intelligent än vad som då var möjligt att skapa. Han kunde med hjälp av WoZ låta en operatör styra systemet som om det vore intelligent, och därmed undersöka hur en människa skulle agera på ett liknande framtida system. Att WoZ effektivt kan simulera framtida hypotetiska scenarion inom MRI gör att det är och troligen fortsatt kommer vara en relevant metod inom området.

2.3 Virtual reality

VR är en teknik som simulerar verklighetstrogna digitala miljöer som en användare kan interagera med [27]. Denna upplevelse möjliggörs genom hårdvara särskilt utvecklad för VR, såsom VR-headset med inbyggda skärmar och linser som simulerar djupseende, sensorer som mäter position i rummet, samt handkontroller med rörelsesensorer som registrerar händernas position och rotation [28]. När användaren rör på huvudet eller kroppen uppdateras synfältet i realtid för att förstärka illusionen av att befinna sig i en verklig miljö, och handkontrollerna möjliggör att användarens armrörelser visualiseras i den virtuella miljön. Knappar och sensorer på handkontrollerna tillåter ofta användaren att på olika vis interagera med digitala objekt. Dessa komponenter bidrar till en immersiv upplevelse där användaren upplever större närvaro i den virtuella världen jämfört med traditionella skärmar och kontroller [27].

Under de senaste åren har det skett en kraftig tillväxt inom VR-marknaden [29] och i dagsläget används VR till allt från underhållning till arbete. Tekniken har även fått stor användning inom områden som design, utbildning och sociala interaktioner där simulering av verklighetstrogna 3D-miljöer kan vara till nytta [27]. I detta projekt är möjligheten till visuell återkoppling från robotens kameror med upplevt djup, tillsammans med intuitiv kontroll, viktiga egenskaper som gör VR till en lämplig teknik.

2.4 Nätverk

En mycket väsentlig del i projektet är kommunikationen mellan enheterna. Kommunikationen sker över nätverket och kräver grundläggande förståelse för OSI-modellen (Open System Interconnection) [30]. OSI-modellen är en teoretisk modell som används för att beskriva hur data kommuniceras i ett nätverk [31]. Genom att dela upp kommunikationsprocessen i sju olika lager kan nätverkskommunikation snabbt och enkelt standardiseras [32]. De sju olika lagren i modellen är:

1. Applikationslagret - Det användaren interagerar med, d.v.s. programmet som körs, till exempel webbläsare.
2. Presentationslagret - Ansvarar för datakonvertering i form av bland annat kryptering och formatering.
3. Sessionslagret - Hanterar kommunikationssessioner mellan två system.
4. Transportlagret - Ansvarar för dataöverföring under en öppen session.
5. Nätverkslagret - Bestämmer hur data ska routas, d.v.s vilken väg data skickas genom nätverket.
6. Datalänklagret - Hanterar nod till nod leverans av data och ser till att data är felfri mellan noder.
7. Fysiska lagret - Den faktiska hårdvaran i form av kablar eller signaler som skickar data.

I detta projekt krävs förståelse för applikationslagret, transportlagret och nätverkslagret, vilka beskrivs mer ingående i följande avsnitt.

2.4.1 Applikationslagret

Applikationslagret är det översta lagret i nätverksmodellen OSI och ansvarar för att tillhandahålla tjänster till användarens applikationer [32]. I detta projekt utnyttjas applikationslagret för att definiera hur applikationerna ska kommunicera med varandra över nätverket. De två vanligaste metoderna för nätverkskommunikation är P2P (Peer-to-Peer) och klient-server, där P2P innebär direkt kommunikation mellan enheter och klient-server involverar en server som hanterar förfrågningar från flera klienter. Fördelen med P2P är att ingen mellanhand behövs, anslutningen är direkt och extrautrustning undviks, nackdelen däremot är att manuell portvidarebefordran i routern behöver göras. I en klient-server-uppsättning slipper användaren manuella inställningar i routern, men en server behöver hela tiden köra, vilket är kostsamt och ger överflödig belastning som i sin tur leder till latens [33].

2.4.2 Transportlagret

TCP (Transmission Control Protocol) och UDP (User Datagram Protocol) är de vanligaste protokollen i transportlagret och fyller olika behov beroende på vilken typ av kommunikation som krävs. TCP är att föredra när det är viktigt att all data kommer fram i rätt skick och ordning. Protokollet är pålitligt eftersom korrupt eller

borttappad data automatiskt felkontrolleras eller återsänds för att garantera korrekt leverans [34]. En nackdel med TCP är den ökade overhead som leder till högre latens. UDP är till skillnad från TCP mycket snabbare eftersom inga felkontroller eller återförsändelser sker [35]. Däremot försvinner garantin för att all data verkligen kommer fram, vilket kan leda till paketförluster eller att data anländer i fel ordning. Detta gör UDP mindre pålitligt, men mer lämpat för realtidsapplikationer där snabbhet är viktigare än noggrannhet [32].

2.4.3 Nätverkslagret

Nätverkslagret, det tredje lagret i OSI-modellen, ansvarar för att dirigera datapaket mellan olika enheter i ett nätverk genom att bestämma den mest effektiva vägen. Processen kallas routing och det viktigaste protokollet som denna process nyttjar är IP (Internet Protocol) [36]. IP används för att adressera olika noder och enheter så att de kan identifieras. En viktig aspekt av nätverkslagret är att kunna särskilja mellan lokala och publika IP-adresser. Lokala IP-adresser används inom ett internt nätverk och börjar oftast med "192.168.x.x"eller "10.x.x.x". Dessa adresser är inte direkt åtkomliga från internet. Publika IP-adresser däremot är de som används för kommunikation över internet och tilldelas av internetleverantören [31]. I detta projekt används både publika och lokala IP-adresser.

2.5 Kinematik för rörelsestyrning

Kinematik är den gren av mekaniken som behandlar kroppars rörelse som en funktion av tid, utan att ta hänsyn till de krafter som orsakar rörelsen [37]. Inom robotik används kinematik specifikt för att beräkna position i rummet utifrån en robots inre ledparametrar, exempelvis ledernas vinklar. I följande underrubriker behandlas både framåtriktad och invers kinematik, som används för rörelsestyrning i detta projekt.

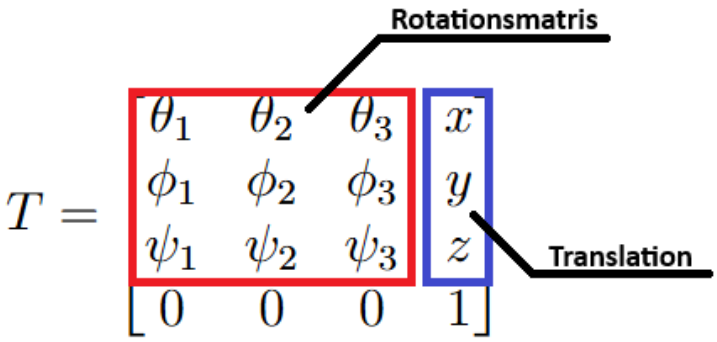
2.5.1 Framåtriktad kinematik

Framåtriktad kinematik används för att beräkna position och orientering av en robots ändeffektor, utifrån information om ledernas längder och rotationsvinklar. En ändeffektor utgör den yttersta länken i en robots arm, vars position man vill hitta i rummet. Positionen ges som en vektor med startpunkt vid fästpunkten för robotens arm och slutpunkt i ändeffektorns centrum. I detta projekt är det specifikt robotens händer som tolkas som ändeffektorer, eftersom deras position är av intresse.

För att bestämma ändeffektorns position modelleras robotens arm som en serie av sammankopplade leder, där varje led associeras med ett lokalt koordinatsystem. I detta projekt definieras varje leds lokala koordinatsystem så att x-axeln pekar i samma riktning som den senaste länken i kedjan. För den första länken i kedjan används ett koordinatsystem som sammanfaller med det globala koordinatsystemet i rummet där ändeffektorns position ges. För att sedan räkna ut ändeffektorns position givet ledernas vinklar och längder, används homogena transformationsmatriser som beskriver förhållandet mellan koordinatsystemen för angränsande leder i serien.

De homogena transformationsmatriserna representerar både rotation och translation jämfört med föregående led i serien.

Den övre vänstra 3x3-delen i transformationsmatrisen, markerad med rött i Figur 2.1, motsvarar en rotationsmatris som beskriver hur axlarnas rotation förändras mellan två leder. Varje kolumn är en enhetsvektor som motsvarar riktningen för x-, y- , respektive z-axeln för nästa länk, uttryckt i den föregående länkens koordinatsystem. 3x1-vektorn i den övre högra delen av matrisen, markerad med blått i Figur 2.1, motsvarar translation längs den förra länkens x-, y- och z-axel. Om en led exempelvis sträcker ut sig längs x-axeln kommer motsvarande sträcka finnas i det översta högra hörnet av matrisen. Eftersom x-axeln är definierad att peka åt samma håll som den senaste länken i kedjan, sker de flesta translationer, bland annat för över- och underarmarna, i den axeln. Den fjärde raden i transformationsmatrisen innehåller tre nollor och avslutas med en etta. Dessa siffror har ingen specifik betydelse, utöver att matrisen ska behålla sina homogena egenskaper.



Figur 2.1: Transformationsmatrisernas struktur.

Eftersom Peppers leder enbart roteras runt en axel i taget, representerar även varje transformationsmatris enbart rotation runt en axel. På grund av detta följer alla rotationsmatriser i projektet en standardiserad modell baserat på vilken axel rotationen sker runt, se Figur A.1. Vinkeln θ motsvarar den aktuella ledens rotationsvinkel, där positiv vinkel representerar moturs rotation och negativ vinkel medurs rotation. Variablerna x, y och z motsvarar translation längs x, y respektive z-axeln. Samtliga homogena transformationsmatriser som används för beräkningen av Peppers framåtriktade kinematik finns listade i Figur A.2. Data angående längden för varje länk i Peppers armar samt rotationsmatriserna kommer från ett tidigare projekt utvecklat av Yuki Suga vid SugarSweet Robotics [38]. För att slutligen få ut ändeffektorns position och rotation i rummet slås alla 7 ihop med matrismultiplikation. Resultatet blir en ny transformationsmatris där rotationer och translationer uttrycks i det ursprungliga koordinatsystemet. Likt transformationsmatriserna för de olika lederna motsvarar den övre vänstra 3x3-delen handens rotation och den övre högra 3x1-vektorn motsvarar position i det globala koordinatsystemet.

2.5.2 Invers kinematik

Medan framåtriktad kinematik är en användbar metod för att räkna ut Peppers handposition utifrån givna vinklar, behövdes även en metod för att lösa det omvända problemet, att få ut vinklar utifrån en önskad position i rummet. Metoden kallas för invers kinematik (IK) och det finns ett flertal olika sätt att implementera den, alla med olika fördelar och nackdelar. De allra flesta metoderna handlar om att antingen lösa ekvationen analytiskt eller iterativt.

En analytisk lösning innebär att man härleder en funktion f som löser ekvationen $f(\vec{v}) = \theta$, där $\theta = [\theta_1, \theta_2, ..., \theta_n]$ är de ledvinklar som krävs för att ändeffektorn ska nå en viss position $\vec{v}$. Denna funktion kan härledas på olika sätt, exempelvis genom att räkna ut inversen till den framåtriktade kinematikens funktion eller genom geometriska observationer. Fördelen med en analytisk lösning är att beräkningen är mycket snabb och resultatet blir exakt, vilket gör det lämpligt för realtidslösningar. Problemet med lösningen är dock att det bara fungerar för robotar med viss struktur samt att det är svårt att hitta en härledning för f för komplexa system.

En iterativ lösning bygger istället på att man med olika metoder iterativt estimerar bättre och bättre lösningar tills man hittar en som är tillräckligt nära positionen man vill nå. Framåtriktad kinematik används i varje iteration för att avgöra hur nära den önskade positionen man är. Det finns flera olika metoder för att göra kvalificerade estimeringar, men gemensamt för de flesta är att man utnyttjar egenskaperna hos en jacobimatris. En jacobimatris är en linjär approximation av en deriverbar funktion nära en given punkt [39, 40] och definieras i detta projekt som:

$$J(\theta) = \left(\frac{\partial f_i}{\partial \theta_j}\right)_{i,j} = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \cdots & \frac{\partial f_1}{\partial \theta_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_i}{\partial \theta_1} & \cdots & \frac{\partial f_i}{\partial \theta_j} \end{bmatrix} \tag{2.1}$$

där $f(\theta) = \begin{bmatrix} x(\theta) & y(\theta) & z(\theta) \end{bmatrix}^T$ är ändeffektorns position i rummet som funktion av ledvinklarna θ (i detta fall, funktionen för framåtriktad kinematik). I detta projekt arbetar vi med en ändeffektor i 3 dimensioner och Peppers armar som har 5 frihetsgrader. Därför har den resulterande jacobimatrisen dimensionen 3x5. Varje kolumn j i jacobimatrisen beskriver en approximation för hur ändeffektorns position förändras när ledvinkel θ_j ändras [39]. Genom att multiplicera jacobimatrisen med en viss vinkelskillnad $\Delta\theta$ får man en approximering för hur stor skillnad det skulle få på ändeffektorns position: $\Delta\vec{v} = J(\theta)\Delta\theta$ [39]. Från ekvationen kan man härleda ett uttryck för en approximativ vinkelskillnad, givet en skillnad i ändeffektorns position:

$$\Delta\theta = J(\theta)^{-1}\Delta\vec{v} \tag{2.2}$$

Eftersom jacobimatrisen inte är kvadratisk, saknar den en vanlig invers. För att ändå kunna lösa ekvationssystemet används Moore-Penrose-inversen, även kallat pseudoinversen. Moore-Penrose-inversen är en generalisering av matrisinvertering som fungerar för alla matriser [41]. Pseudoinversen för jacobimatrisen ges av följande formel:

$$J^+ = J^T(JJ^T)^{-1} \tag{2.3}$$

I vissa fall kan det hända att pseudoinversen är instabil, exempelvis om jacobimatrisen är singulär [41], [42]. Därför används även en dämpningsterm λ för att stabilisera inversen och se till att den aldrig blir singulär [42]. Detta görs genom att addera en matris med dämpningstermen i kvadrat på diagonalen, och leder bland annat till att känsligheten för brus och avrundningsfel minskar. Tillämpningen av en dämpad pseudoinvers sker enligt följande formel:

$$J_\lambda^+ = J^T(JJ^T + \lambda^2 I)^{-1} \tag{2.4}$$

På grund av att jacobimatrisen endast representerar en linjär approximation runt en punkt blir resultatet från Ekvation 2.2 inte exakt, och ju större skillnad i ändeffektorns position, desto större blir felet. Därför används en dämpningsfaktor för att minska påverkan från jacobimatrisen. Beräkningen sker iterativt, där varje iteration närmar sig önskad position tills en acceptabel felmarginal uppnås. Eftersom det beroende på vald felmarginal kan ta många iterationer innan en lösning hittas, blir metoden långsammare än en rent analytisk lösning. Metoden är också känslig för dåliga initialgissningar, men detta kan motverkas genom att ständigt använda de senaste vinklarna som initial gissning. Några fördelar är att metoden är flexibel och fungerar på alla robotkonfigurationer. Det är även möjligt att implementera metoder för att hantera redundans eller begränsningar för ledvinklar, vilket är svårare i en analytisk lösning.

2.5.3 Vald implementering av IK

I detta projekt används en iterativ lösning för invers kinematik. Eftersom spårningen av operatörens armar är begränsad till handpositioner och armbågen kan rotera i flera plan, är det svårt att hitta en analytisk lösning som klarar alla Peppers möjliga rörelser. Peppers ledvinklar har dessutom begränsningar som måste följas, varför en iterativ lösning passar bättre. Lösningen följer till stor del grundläggande implementeringar från tidigare avsnitt, men innehåller även några modifikationer.

Den största modifikationen är att förutom beräkning av delta-vinklar sker en sekundär uppgift som arbetar för att lösningen hålls inom Peppers ledgränser. Om lösningen för någon ledvinkel hamnar nära gränserna förs vinklarna bort från dem via en nollrumsprojektion. En nollrumsprojektion innebär att endast de komponenter av vinklarna som inte påverkar rörelse av ändeffektorn mot slutpunkten uppdateras. Nollrumsprojektionen ser till att vinklarna förs bort från gränserna, förutsatt att det inte påverkar möjligheten att nå önskad position.

En annan modifikation är en speciell dämpningsfaktor som används för att förhindra svängningar i beräkningarna. Om en gissning visat sig vara sämre än föregående gissning blir dämpningen starkare för att det inte ska börja oscillera fram och tillbaka. Samma dämpningsfaktor används även för att avgöra om beräkningarna fastnat i ett lokalt minimum. När beräkningarna fastnar i ett lokalt minimum blir dämpningen starkare för varje iteration, och när skillnaden mellan iterationerna blir för liten startar algoritmen om med nya startvinklar. Detta upprepas tills ett max antal iterationer uppnåtts för att öka chansen att lösningar hittas utan att låsa systemet om det inte finns någon lösning.

2.6 NARS

NARS (Negative Attitude toward Robot Scale), är ett formulär som togs fram av [43] för att kunna mäta negativa attityder mot robotar. NARS använder sig av en likertskala, vilket är en skala från 1-5 där deltagaren får gradera hur mycket en fråga passar in på dem. Detta formulär gör deltagaren innan interaktionen med roboten för att få en förståelse för deltagarens attityd mot robotar. Formuläret fokuserar på tre delskalor:

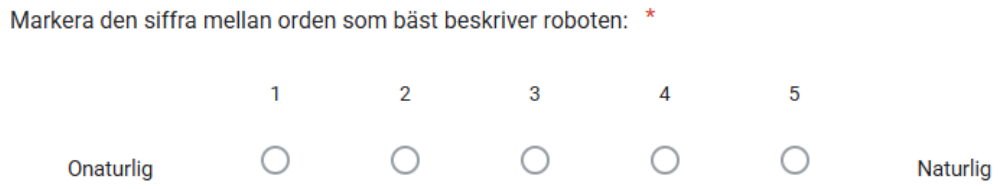
- **S1 Negative attitude toward interaction with robots:** Negativa känslor inför att interagera med robotar.
- **S2 Negative attitude toward social influence of robots:** Rädsla för robotars potentiella påverkan på samhälle och kultur.
- **S3 Negative attitude toward emotional interactions with robots:** Skepsis mot att utveckla emotionella band till robotar.

Frågorna som visas i Tabell B.2 är översatta från den engelska versionen som togs fram av [43], som har översatts och validerats enligt [44]. Standarden för NARS är att 5 är svaret med mest negativ attityd, undantaget är de inverterade frågorna som istället följer motsatsen. Inverterade frågor är markerade i Tabell B.2 med *.

2.7 GODSPEED

Det fanns länge ett behov av standardiserade och tillförlitliga sätt att utvärdera hur människor upplevde interaktioner med robotar. GODSPEED är ett formulär som togs fram av [45] för detta syfte, som likt NARS, blivit översatt och validerat av [44]. Formuläret är strukturerat i form av fem kategorier, där respektive kategoris frågor likt NARS använder en likertskala, men här en skala mellan två motsatsord, vilket senare resulterar i ett snittvärde per kategori. Ett exempel på hur en fråga ser ut visas i Figur 2.2. De fem kategorierna som ingår i GODSPEED är:

- **1 Anthropomorphism:** Människolikhet
- **2 Animacy:** Livfullhet
- **3 Likeability:** Trevlighet
- **4 Perceived Intelligence:** Upplevd intelligens
- **5 Perceived Safety:** Upplevd säkerhet



Figur 2.2: Exempel på fråga i GODSPEED.

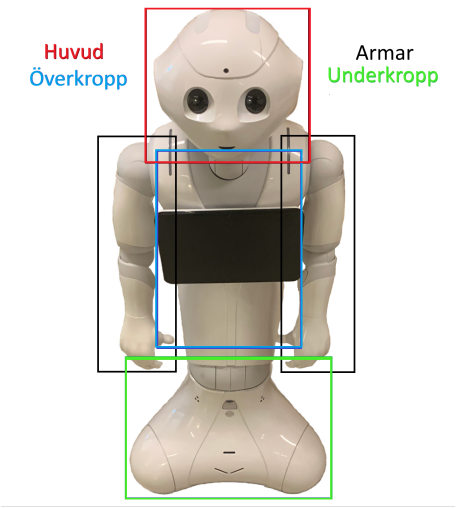
3

Hårdvara och mjukvara

I detta kapitel presenteras den hårdvara och mjukvara som utgör systemets grund. Kapitelinnehållet omfattar en beskrivning av robotens fysiska komponenter samt en genomgång av den programvara som använts under projektets gång.

3.1 Roboten Pepper

Pepper är modulärt uppbyggd, vilket innebär att varje enskild kroppsdel kan styras oberoende av de övriga. Till följd av detta har huvud, armar, överkropp och manövreringsplattform var och en sina egna begränsningar för rörelse och rotation, vilka är konstanta oavsett andra komponenters positionering. Trots självständigheten inom konstruktionen är samtliga komponenter i roboten anslutna till ett övergripande nätverk av sensorer, vilket aktivt övervakar och övertar kontrollen av roboten vid behov. Sensorernas huvudsakliga ändamål är att förhindra kollisioner, överansträngning av motorer och fall vid obalans. Utöver de dynamiska delarna av roboten är Pepper även utrustad med kompletterande hårdvara i form av bland annat ljuskällor, kameror och mikrofoner [25]. I kommande avsnitt ges en översiktlig beskrivning av komponenter hos Pepper relevanta för detta arbete.



Figur 3.1: Uppdelning av Pepper i fyra sektioner.

3.1.1 Programmering: Python och NAOqi

NAOqi är benämningen på det inbyggda programvarusystemet som driver Pepper. Ett ramverk skapat av företaget Aldebaran Robotics som förser användare med en strukturerad miljö för att utveckla programvara utnyttjandes hanteringen av ljud, sensorer, motorer och kameror [46]. Ramverket är plattformsoberoende, vilket gör det möjligt att utveckla för både Windows, macOS och Linux. Python, C++, Java, JavaScript, och ROS stöds som programmeringsspråk [47]. I detta projekt valde vi Python på grund av dess enkla utformning, gruppens erfarenhet av språket samt kompatibiliteten med programvaran Choregraphe [48], som använts för simulering. Tabell 3.1 redovisar de olika programmeringsspråkens kompatibilitet.

Tabell 3.1: Tabell med kompatibiliteten hos Pepper för olika programmeringsspråk.

Programmeringsspråk	Bindningar för		Choregraphe	
Kompatibel = Ja	Dator	Robot	Bygga applikationer	Kod redigering
Python	Ja	Ja	Ja	Ja
C++	Ja	Ja	Nej	Nej
Java	Ja	Nej	Nej	Nej
JavaScript	Ja	Ja	Ja	Nej
ROS	Ja	Nej	Nej	Nej

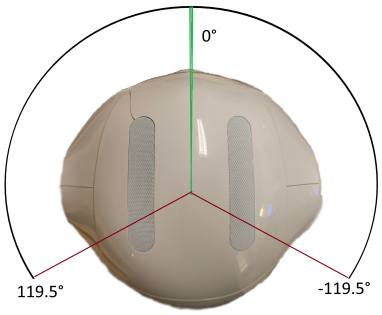
Projektets programmering för Pepper förlitar sig starkt på det relaterade Python SDK:t, vilket genom NAOqi:s API översätter enklare kommandon till rörelser hos roboten. Funktioner från det importerade biblioteket möjliggör styrning genom enklare inmatning av vinklar, koordinater och hastigheter. Liknande funktioner för hämtning av videoflöde, sensordata, ljud och position i robotens interna koordinatsystem existerar även inom biblioteket [49]. Eftersom Python SDK:t endast kompilerats för Python 2.7 32-bit, används Python 2.7.18 32-bit i detta projekt.

3.1.2 Huvud

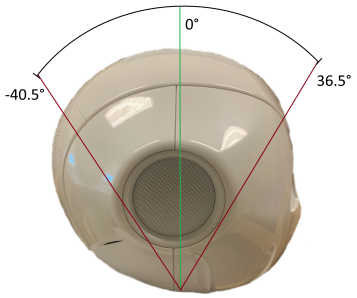
Huvudet av Pepper har två frihetsgrader, *HeadYaw* (rotation i sidled), och *HeadPitch* (lutning upp och ner). *HeadYaw* tillåter en rotation på $\pm 119,5°$, där 0° innebär ingen vridning från bröstkorgens riktning. *HeadPitch* kan ge en lutning av huvudet mellan +30° till -40,5°, där 0° är parallellt med ryggraden [50]. Nedan illustreras rörelser inom rotationsintervallen i Figur 3.2 och Figur 3.3.

3.1.3 Överkropp

Överkroppen hos Pepper är konstruerad för att möjliggöra viss rotation från höftleden, både i sidled (*roll*) och lutning framåt (*pitch*) [50]. I detta projekt utnyttjas dock inte denna funktionalitet genom direkt styrning. Istället används robotens inbyggda balanseringssystem för att kompensera rörelser som uppstår från styrsystemet. Detta möjliggörs av NAOqi-ramverkets inbyggda rörelsemodul `ALMotion`, som använder data från sensorer för att justera överkroppen [51].



Figur 3.2: HeadYaw.



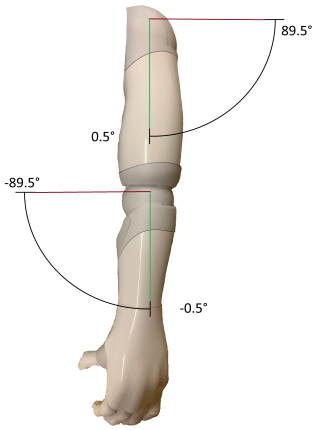
Figur 3.3: HeadPitch.

3.1.4 Armar

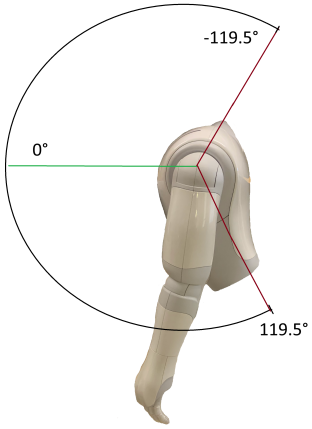
Peppers armar är modulärt uppbyggda av flera segment. Mer specifikt består varje arm av tre segment, överarm, underarm och hand [50]. I detta projekt styrs fyra rotationsaxlar per arm, två från vardera axel och armbåge:

- **Axeln** (styr överarm):
  - Rotation i sidled (*Shoulder roll*)
  - Rotation upp och ned (*Shoulder pitch*)
- **Armbågen** (styr underarm):
  - Böjning inåt (*Elbow roll*)
  - Rotation kring underarmens egna axel (*Elbow yaw*)

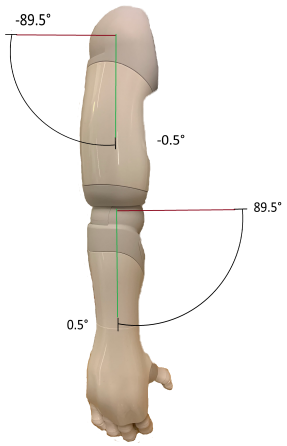
Utöver de ovanstående finns ytterligare en led för rotation i handleden (*wrist yaw*), vilken i detta projekt inte används i samma utsträckning. Under större delar av styrningen tillåts istället handleden huvudsakligen följa med i underarmens rotation (*elbow yaw*). Roboten har även motorer för att öppna och stänga sina händer, där individuell kontroll av enskilda fingrar inte är möjlig. Visualisering av ledernas rörelser, samt deras rotationsbegränsningar för vardera arm presenteras i Figur 3.5-Figur 3.9 nedan, där indikation för vilken arm som illustreras benämns genom 'R' (högra) alternativt 'L' (vänstra) före namnet på rotationspunkten.



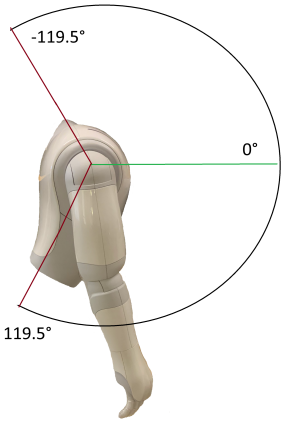
Figur 3.4: LShoulderRoll, LElbowRoll.



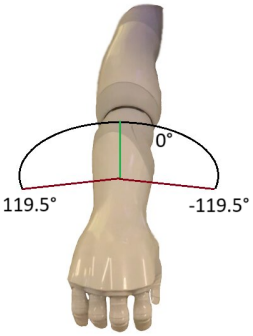
Figur 3.5: LShoulderPitch.



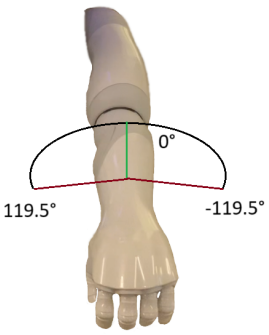
Figur 3.6: RShoulderRoll, RElbowRoll.



Figur 3.7: RShoulderPitch.



Figur 3.8: RElbowYaw.



Figur 3.9: LElbowYaw.

3.1.5 Underkropp

Peppers underkropp hanterar förflyttelse inom rummet och är utrustad med tre individuellt styrda sfäriska hjul [52]. Detta innebär att den kan förflytta sig fritt i valfri riktning längs golvet, oavsett rotationsriktning på kroppen, och möjliggör en full rotation i två riktningar [53]. I detta arbete används underkroppen för att återspegla operatörens rörelser genom att mäta förflyttningar inom en spelmiljö som sedan omvandlas till motsvarande motorhastigheter för vardera hjul, användandes en rörelsemodul inom NAOqi benämnd `ALMotion` [54].

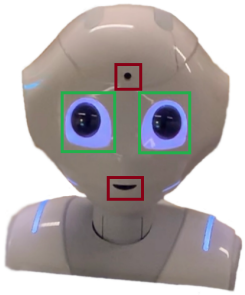
Figur 3.10: Peppers bas framifrån.

Figur 3.11: Peppers bas från sidan.

3.1.6 Kameror

Pepper är utrustad med fyra kameror monterade i huvudets hölje, i pannan, munnen samt i vardera öga. Kamerorna i pannan och vid munnen fungerar som vanliga 2D-kameror, medan ögonkamerorna är placerade och utformade för att generera en stereoskopisk bild [55], [56]. Genom att kombinera bildinformationen från båda ögonen kan utvecklare åstadkomma djupseende.

I detta projekt används enbart ögonkamerorna med deras stereoskopiska konfiguration. Detta eftersom djupuppfattning ansågs vara av stor vikt för en realistisk VR-upplevelse.



Figur 3.12: Placering av kameror.

Figur 3.13: Visualisering av synfält.

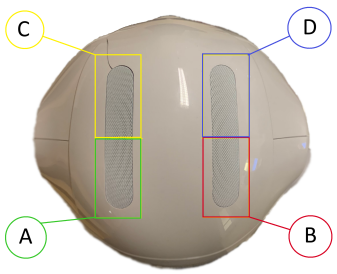
3.1.7 Sensorer

Pepper är utrustad med flera inbyggda sensorer som används för att stödja rörelse, balans, interaktion och motverkan av överansträngning [57]. Bland dessa sensorer är det däremot tre huvudsakliga grupperingar som spelar en mer aktiv roll i detta arbete.

- **IMU (Inertial Measurement Unit):** Bröstkorgen av Pepper innehåller varsin 3-axis gyrometer och accelerometer [58]. Dessa används av robotens interna balanseringssystem för att uppskatta och kompensera rörelser utförda under styrningen.
- **Kontakt- och känselsensorer:** Pepper är utrustad med taktila sensorer på huvudet, händerna samt genom stötfångare på manövreringsplattformen [59]. Dessa sensorer reagerar på fysisk kontakt och är huvudsakligen avsedda för att skydda hårdvaran. Sensorerna kan automatiskt utlösa beteendeavbrott för att undvika kollisioner eller tyngre påfrestning. I interaktiva scenarier, som i detta arbete, kan sensorerna leda till oönskade avbrott och programlogisk förvirring. Exempelvis kan styrning under ett avbrott köas upp och först verkställas när roboten återgår till ett aktivt tillstånd.
- **Sonar- och infraröd:**
  Närmare basen av Pepper finns två sonar- och infrarödsensorer [60], [61]. Dessa sensorer används för att mäta avstånd och upptäcka främmande objekt i robotens omgivning, vilket hjälper till att förhindra kollision med väggar och hinder. Sensorerna bidrar till att skydda robotens hårdvara, men kan orsaka problem i trånga utrymmen eller vid kontakt med människor. Likt de taktila sensorerna orsakar dessa beteendeavbrott, och därav ett mindre naturligt samt responsivt beteende hos roboten.

3.1.8 Mikrofoner

Pepper är utrustad med fyra mikrofoner placerade på ovansidan av robotens huvud [62]. I detta projekt används mikrofonerna för att lyssna på ljud från omgivningen, exempelvis tal från personer runtom Pepper. Ljudet överförs till operatörens VR-utrustning i realtid, vilket möjliggör responsiv styrning av roboten baserat på auditiva instruktioner.



Figur 3.14: Placering och benämning av mikrofoner.

3.2 Meta Quest 2

I detta projekt används Meta Quest 2 som VR-utrustning [63]. Det är ett av de mest sålda headseten på marknaden [64], vilket gör det till ett relevant val för att möjliggöra bred användning av projektets programvara. Utrustningen består av ett fristående headset och två kontroller. Genom så kallad inside-out tracking används inbyggda kameror för att spåra användarens rörelser i realtid utan behov av externa sensorer eller basstation [65]. Headsetet stödjer dessutom trådlös anslutning till en dator via Meta Air Link [66], vilket gör det möjligt att spegla programvara från datorn utan kablar. Denna frihet ger operatören ökad rörelseförmåga och underlättar styrningen av Pepper i större lokaler.

Figur 3.15: Meta quest 2.

3.3 Unity

All programvara som hanterar VR-utrustningen samt de tillhörande beräkningarna har utvecklats med Unity. Unity är en populär spelmotor som används för att skapa interaktiva applikationer för olika plattformar, inklusive PC, Mac och VR-enheter. I Unity erbjuds en hierarkibaserad utvecklingsprocess där man kan skapa objekt i en 3D-miljö och sedan enkelt koppla dessa objekt till andra objekt eller skript som utför olika funktioner. Det finns även möjlighet för felsökning genom integrerade spel- och konsolfönster.

En av de främsta fördelarna med att använda Unity för VR-utveckling, jämfört med alternativa plattformar och metoder, är att spelmotorn erbjuder en kostnadsfri grundversion samt en intuitiv och användarvänlig utvecklingsmiljö. Därtill är Unity en av de mest etablerade spelmotorerna på marknaden och har en stark position inom VR-utveckling, där cirka 60% av alla VR-spel är skapade med Unity [67]. Detta beror till stor del på det stora stödet som finns för olika VR-standarder och det finns en mängd olika tilläggsmoduler specialutvecklade för VR-utveckling.

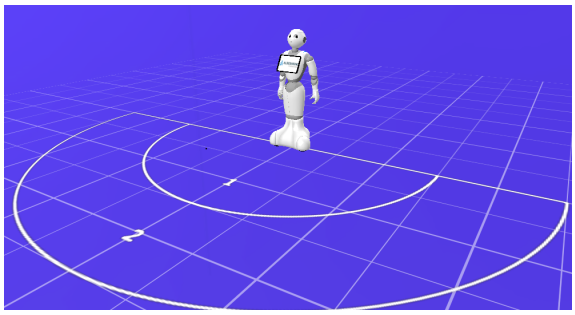
Vi har använt Unity 6 som under arbetets gång var den senaste och mest stabila versionen av programmet [68]. Programmeringsspråket som används för att skriva

skript som ligger till grund för logiken i styrsystemet är C# 9.0. C# är ett objektorienterat språk som lämpade sig bra för de tyngre beräkningarna i projektet eftersom språket är betydligt snabbare än Python. Dessutom finns det extra funktioner och datatyper för C# i Unity, bland annat kvaternioner och vektorer för rotation respektive positionering i en 3D-miljö.

Unity erbjuder även smidig integration med OpenXR som är en öppen standard utvecklad av Khronos Group [69] som skapar ett API-lager för att underlätta utveckling av applikationer för flera VR-enheter. Detta innebär att även om vi primärt utvecklat styrsystemet med Meta Quest 2 i åtanke så kommer det även att fungera med annan VR-utrustning som HTC Vive eller Valve Index eftersom instruktionerna som används är standardiserade. OpenXR-biblioteket erbjuder ett flertal färdiga moduler för exempelvis projicering av 3D-miljön till ett VR-headset samt positionering av headset och kontroller.

3.4 Choregraphe

Choregraphe är en applikation utvecklad av SoftBank Robotics som möjliggör 3D-simulering av virtuella robotar, där bland annat modellen Pepper stöds [48].



Figur 3.16: Simuleringsmiljö i Choregraphe.

Genom att koppla Choregraphe till en virtuell robot via en konsol som kör det medföljande programmet *naoqi-bin.exe*, kan utvecklare visualisera hur Pepper reagerar inom en simulerad miljö. Detta gör det möjligt att testa oprövad kod utan risk för att skada den fysiska roboten. Det underlättar även utveckling och felsökning på distans, vilket är särskilt värdefullt när tillgången till roboten är begränsad.

3.5 IBM SPSS statistics

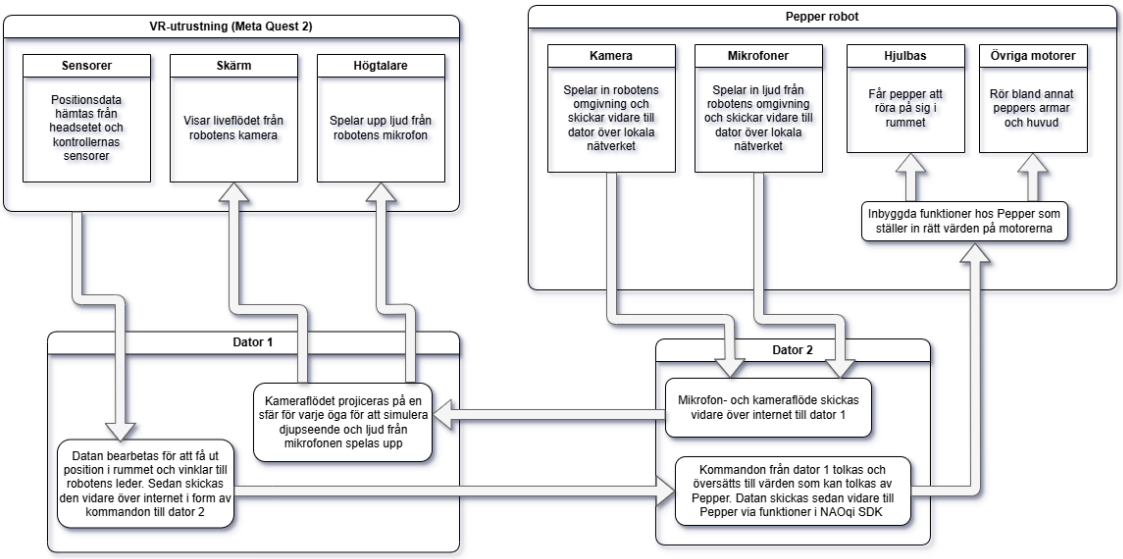
IBM SPSS Statistics 30 är en programvara som används för att statistiskt analysera data. Det är särskilt bra för att effektivt kunna hantera större mängder data. När väl all data är inmatad kan användaren enkelt visualisera resultatet i form av genererade tabeller och grafer, vilket underlättar tolkningen av resultatet. Detta gör SPSS till ett lämpligt verktyg för att analysera och visualisera svar från enkäter.

4
Implementation och utveckling

Detta kapitel redogör för systemets implementation och uppbyggnad, hur kommunikationen sker mellan olika komponenter, samt den övergripande systemarkitekturen.

4.1 Systemarkitektur

Styrsystemet består av fyra huvudkomponenter, Pepper, VR-utrustningen och två datorer. De två datorerna kan befinna sig på olika eller samma nätverk och hanterar databearbetning och kommunikation mellan Pepper och VR-utrustningen. Dator 1 kör ett C#-program skapat i Unity medan Dator 2 kör ett Python-program. För att underlätta felsökning och utveckling har båda programmen en skalbar struktur. Olika funktioner av styrsystemet kan därav smidigt kopplas av och på, och vid händelse av att ett eller flera delsystem kraschar kommer resterande fortfarande att fungera.



Figur 4.1: Övergripande flödesschema för systemarkitekturen.

Som Figur 4.1 visar, hämtar Dator 1 rörelsedata från VR-utrustningen och beräknar vilka instruktioner i form av positioneringsdata som ska skickas till dator 2. Dator 2 hanterar all kommunikation med Pepper och utför vid behov justeringar baserat på robotens aktuella tillstånd för att applicera den nya datan som återspeglar operatörens rörelser på Pepper. Samtidigt hämtar och behandlar dator 2 ljud och bild

från Pepper, vilket sedan skickas vidare till dator 1. Där återskapas ett bildflöde i VR-miljön och ljudet spelas upp från utrustningen.

4.1.1 Datapaket

Datapaketen som skickas över nätverket innehåller instruktioner, ljud och bild. Instruktionsdata skickas från Dator 1 till Dator 2, ljud och bild skickas från Dator 2 till Dator 1. Instruktioner och bilder skickas med transportprotokollet TCP medan ljud skickas med protokollet UDP. För att minimera data som skickas över nätverket och minimera exekveringstiden på programmen har instruktionspaketen delats upp i fem olika meddelanden:

```
quit{     type: quit      }        move{     type: move,
                                              x: value,
head{     type: head,                         y: value,
          HeadYaw: value,                     theta: value,
          HeadPitch: value}                   z: value        }

rightArm{ type: rightArm,          leftArm{  type: leftArm,
          RShoulderPitch: value,             LShoulderPitch: value,
          RShoulderRoll: value,              LShoulderRoll: value,
          RElbowYaw: value,                  LElbowYaw: value,
          RElbowRoll: value,                 LElbowRoll: value,
          RWristYaw: value,                  LWristYaw: value,
          RHand: value    }                  LHand: value    }
```

Figur 4.2: Datastruktur för olika kommandon till roboten.

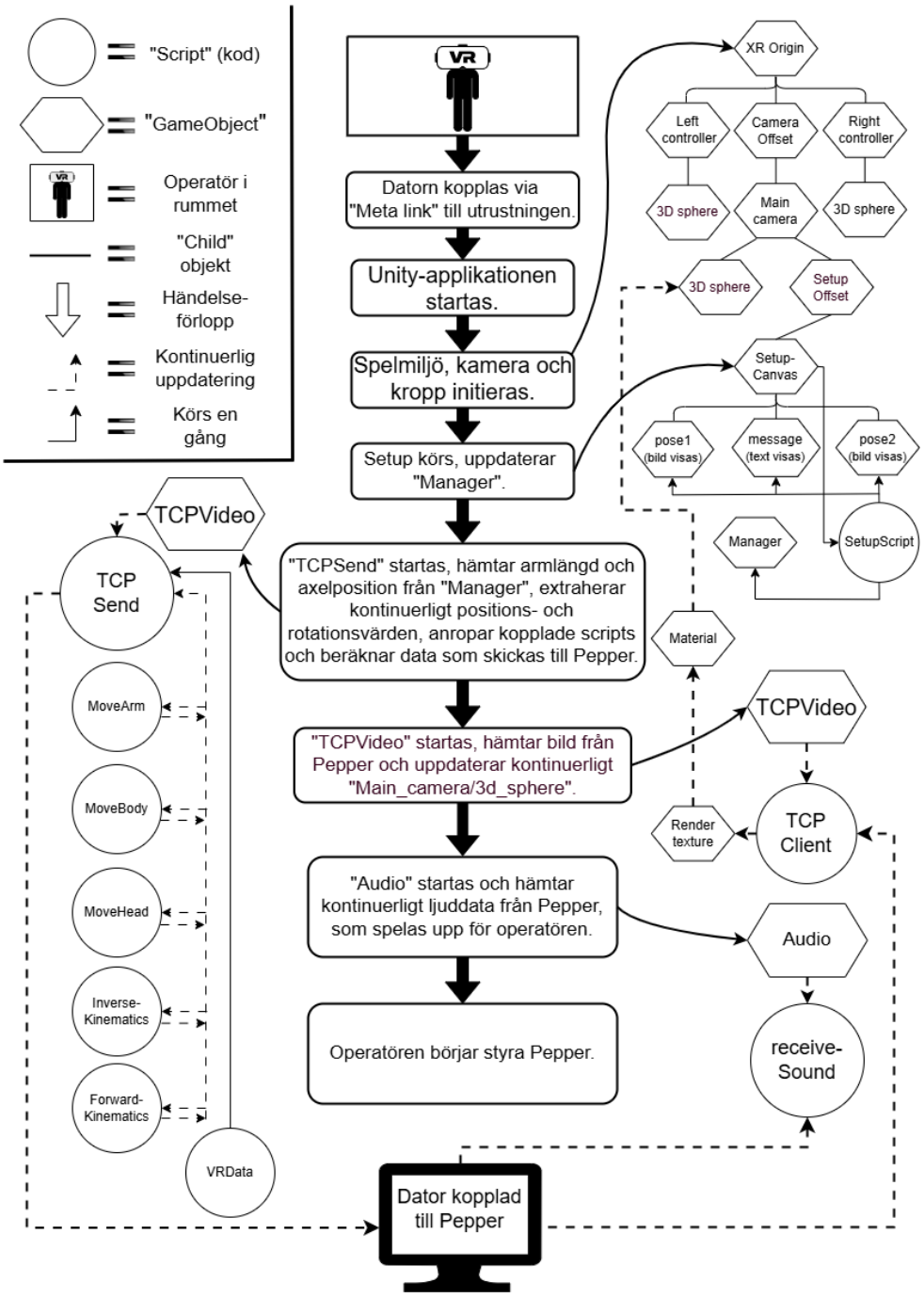
Varje meddelande hanteras separat av systemet, och då de består av varierande mängd parametrar medföljer alltid storleken av inkommande meddelande. Frekvens av nya meddelanden ställer operatören själv in genom reglering av systemparametrars gränsvärden, exempelvis avståndet från operatörens nuvarande position som anses stort nog för att uppdatera robotens. Lägre gränser, och därav högre frekvenser ger ökad precision för systemet, men även en större belastning på nätverk och hårdvara.

Bild från Pepper hämtas med en upplösning på 640x180 och önskad uppdateringsfrekvens på 15 FPS. På grund av Peppers interna processorkraft, kan däremot inte en 15 bilder per sekund garanteras. Bilderna tolkas i RGB format, vilket innebär att varje pixel behöver tre bytes för att representera hela färgskalan. Innan bilden skickas komprimeras den till JPEG format vilket minskar bildens filstorlek och därmed belastningen på nätverket.

Ljudpaketen skickas över UDP och har en samplingsfrekvens på 16 kHz. Frekvensen är lämpligt för tal men kommer inte fungera optimalt i miljöer där mer detaljerad kvalitet eftersöks. Pepper har en ljudbuffert på 170 ms, vilket i teorin innebär att informationen skickas över nätverket med ett intervall på 170 ms. Hur ofta ljudpaketen skickas i praktiken däremot beror på Peppers interna schemaläggning, vilket påverkas av antalet aktiva processer.

4.2 Unity - Dator 1

För att möjliggöra styrning av Pepper med hjälp av VR används Unity för att skapa en spelapplikation som omvandlar operatörens rörelser till applicerbar data. För att uppfylla detta ansvarsområde hanterar applikationen även visuell och auditiv återkoppling vilket möjliggör adaptiv styrning i realtid. En förenklad övergripande bild av applikationens händelseförlopp presenteras i Figur 4.3.



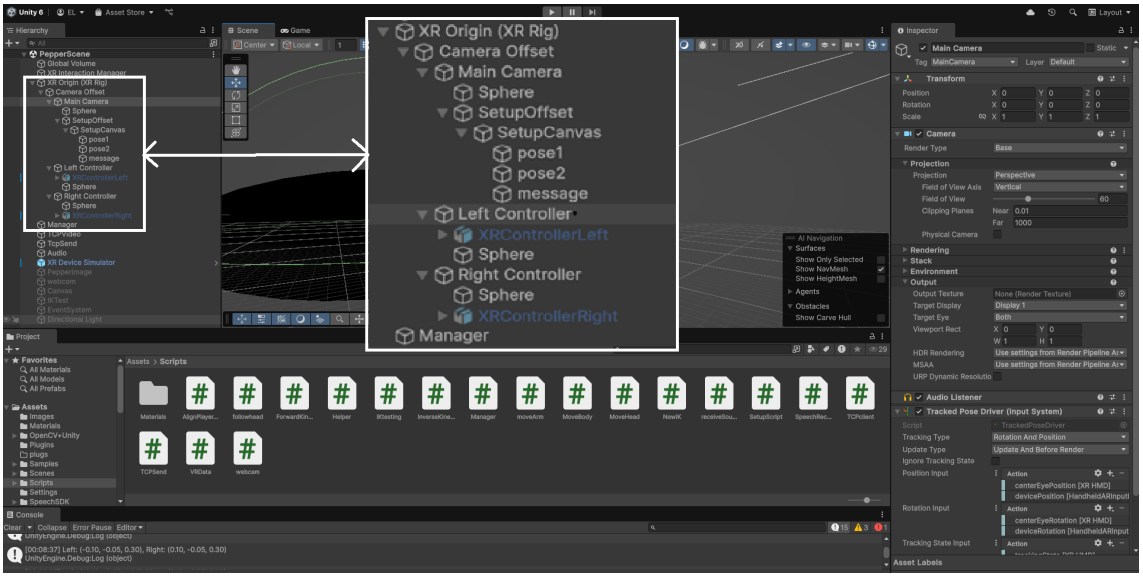
Figur 4.3: Hierarki av Unity.

Användandes den inbyggda programvaran Meta Air Link från Meta Quest 2 kopplas VR-utrustningen till Dator 1. Vid start av applikationen initialiseras spelmiljön för att sedan påbörja kalibrering genom en fas refererad till som Setup i detta arbete. Under Setup får användaren instruktioner om att korrigera sin kroppshållning efter illustrerade bilder. Datan från kalibreringen används för att uppskatta operatörens armlängd och axelposition, vilka sparas i programmet genom ett `gameobject` (spelobjekt) benämnt `Manager`, som agerar i form av variabelhållare.

Efter `Setup` påbörjar applikationen de tre centrala huvudmodulerna som hanterar rörelse, visuell miljö och ljud. `TCPsend`, benämnt efter att vara den enda exporterande modulen, ansvarar för att med hjälp av ett flertal stödjande skript hämta ut positions- och rotationsdata från spelvariabler, vilket efter bearbetning skickas vidare genom en TCP-anslutning. Den parallella modulen `TCPVideo` tar emot bilddata via en separat TCP-anslutning som kontinuerligt projiceras på insidan av en 3D-sfär med användaren centrerad inuti. Detta resulterar i en “live video“ som omsluter operatören för att ge en mer immersiv upplevelse. För att möjliggöra ljud ansluts ett spelobjekt vid namn `Audio` till en UDP-anslutning, vilket säkerställer att mottagen ljuddata från Peppers omgivning spelas upp från VR-headsetet. Till skillnad från kalibreringen och initialiseringen av applikationen körs dessa tre moduler kontinuerligt under den aktiva styrningen. De olika delsystemens struktur redovisas i ytterligare detalj under följande sektioner.

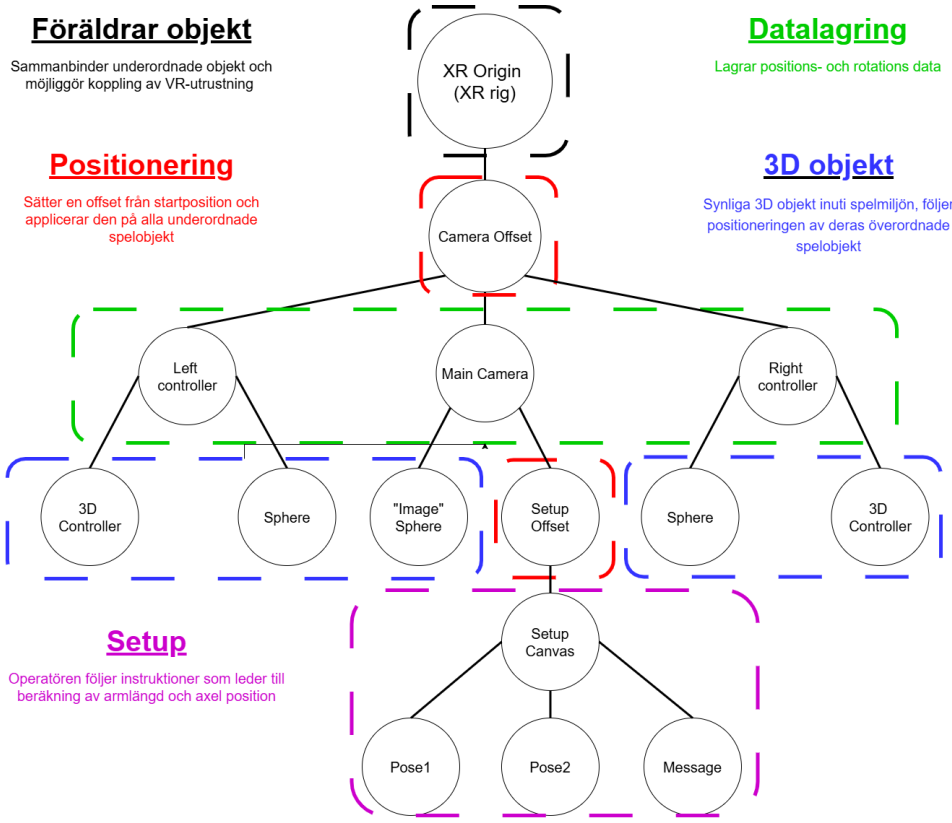
4.2.1 Initialisering och Setup

Innan aktiv styrning påbörjas måste initialiseringen av spelmiljön och kalibreringen av användarens axlar och armar genomföras. Stegen som utgör denna förberedelsefas beskrivs nedan, tillsammans med en hänvisning till relaterade avsnitt inom Unity-applikationens hierarki i Figur 4.4.



Figur 4.4: Hierarki avsnitt relaterat till initialisering och Setup.

Initialisering syftar till att upprätta speglingen mellan operatörens utrustning och spelvärldens "avatar". I detta projekt används den inbyggda komponenten `XR Origin (XR Rig)` från Unitys egen *XR Interaction Toolkit*. Spelobjektet och dess underordnade komponenter omhändertar registreringen av operatörens förflyttningar, rotationer och knapptryck till korresponderande positionering av avataren inuti spelmiljön. Position och rotation för huvud och händer går därefter att observera inuti de underordnade spelobjektens transformer. Figur 4.5 visar spelobjektet `XR Origin (XR Rig)` och dess underordnade spelobjekt med en övergripande förklaring.



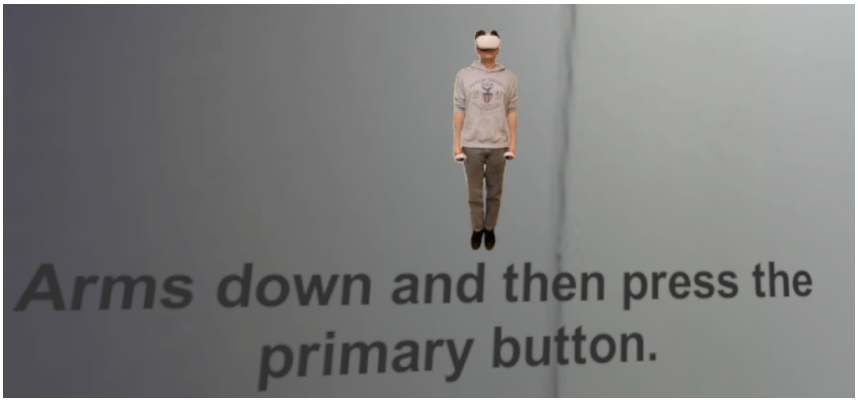
Figur 4.5: Hierarki av XR Origin och dess underordnade spelobjekt.

Först i hierarkins ordningsföljd efter `XR Origin` är `Camera Offset`, vars funktion är att förskjuta positioneringen av alla underordnade objekt. `Main Camera` är direkt kopplad till operatörens VR-headset och använder en inbyggd *Tracked Pose Driver* för att rotera och förflyttas i enlighet med operatören. Genom kontinuerlig uppdatering används detta spelobjekt som källa för extrahering av positions- och rotationsdata gällande huvudet till andra program inom applikationen. `Main Camera` har även två egna underordnade spelobjekt i form av en större ihålig sfär och `Setup Offset`. Som underordnade objekt förblir sfären och *Setup Offset* centrerad kring positionen av `Main Camera` oavsett rörelse eller rotation. Sfären används senare i applikationens händelseförlopp som en duk för projicering av videoflödet med hjälp av `TCPVideo`, samtidigt som `Setup Offset` används för att distansera `SetupCanvas` ett konstant avstånd framför operatörens vy.

De resterande två underordnade objekten till `Camera Offset` är `Left Controller` och `Right Controller`. Likt `Main Camera` är dessa kopplade till VR-utrustningen, men speglar istället händernas rörelse och rotation. Både vänster och höger kontroller har ytterligare underordnade objekt som möjliggör visuell representation av händerna i 3D-miljön, operatören får själv göra ett val mellan animerad VR-kontroller eller enklare sfär för en mer simplistisk design.

`SetupCanvas` placeras framför operatören baserat på det fördefinierade avståndet `SetupOffset`. Spelobjektet agerar som en visuell duk där bilder och instruktioner kan illustreras. `SetupCanvas` har tre underordnade spelobjekt, två bilder, `Pose1` (se Figur 4.6) och `Pose2` (se Figur 4.7), samt en textruta kallad `Message`. Kopplat till `SetupCanvas` är skriptet *SetupScript.cs* som omhändertar logiska operationer, bytet av bilder, uppdateringen av meddelanden och registrering av knapptryck.

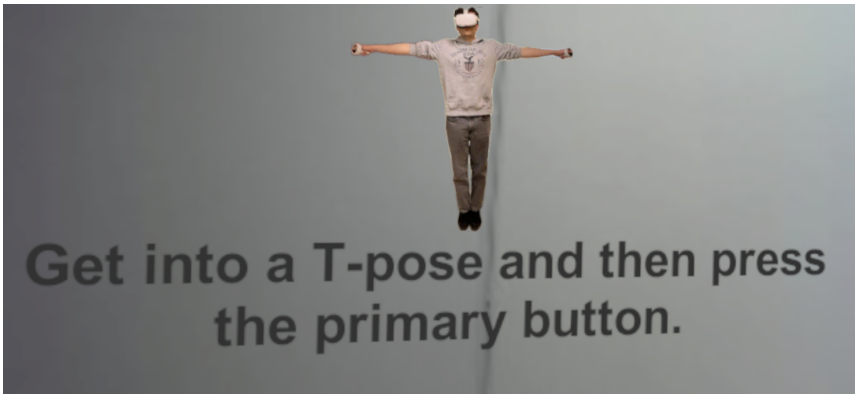
Vid initiering av *SetupScript.cs* visas bilden `Pose1` tillsammans med ett meddelande som ber operatören inta en neutral ställning med armarna längs sidan av kroppen. I denna kroppshållning placeras händerna i samma x- och z-koordinat som respektive axel, medan endast y-koordinaten (höjdnivån) skiljer sig mellan de två. När operatören är redo instrueras hen att trycka på knappen *Primary button*, vilket kallar på funktionen `OnConfirmArmsDownPose()`. Denna funktion roterar spelavataren kring y-axeln så att sidledsrörelser enbart sker längs x-axeln, respektive framåtriktade rörelser längs z-axeln. Därefter sparas positionsskillnaden mellan vardera hand och huvud i tre dimensioner och den visade bilden uppdateras till `Pose2` med ett nytt meddelande.



Figur 4.6: Skärmbild av operatörens vy under Setup, med Pose1.

`Pose2` instruerar operatören att lyfta armarna i sidled till axelhöjd (T-pose), för att sedan trycka på *Primary button*. Denna kroppshållning innebär att skillnad av y-koordinaten från huvudet till handen, blir lika långt som från huvudet till axeln. Efter knapptrycket kallas funktionen `OnConfirmTPose()`, som utför en nollställning av spelavatarens y-rotation, initialiserar vektorer för axlarnas position, estimerar armarnas längd, och sparar y-koordinaten av händerna i en variabeln nämnd `shoulderHeight`. Vektorerna `leftShoulderDelta` och `rightShoulderDelta` utformas genom att kombinera x- och z-värdena från händernas position i `Pose1`, med

y-värdena från `Pose2`. Armlängd, `armLength`, beräknas som medelvärdet av skillnaderna i y-position för respektive arm mellan de två kroppshållningarna. När all nödvändig data har beräknats skickas de till spelobjektet `Manager`. Därefter visar `SetupCanvas` ett avslutande meddelande och spelobjektet `TCPSend` aktiveras. Avslutningsvis kallas funktionen `DestroySelf()`, som förstör `SetupCanvas`.



Figur 4.7: Skärmbild av operatörens vy under Setup, med Pose2.

4.2.2 TCPVideo

Spelobjektet `TCPVideo` användande dess kopplade skript *TcpVideoReceiver.cs*, omhändertar den visuella återkopplingen från Peppers kameror. Bilddata skickas i realtid genom en etablerad TCP-anslutning från datorn kopplad till Pepper. För att undvika att applikationen fryser i väntan på nya meddelanden, eller vid bearbetning av data, sker nätverkskommunikation och logiska operationer i en separat *Thread* (tråd). Däremot kräver Unity att uppdatering av visuella element, i detta fall UI-komponenter, sker inom applikationens huvudtråd *MainThread*.

Bakgrundstråden berör huvudsakligen funktionen `CreateImage()` som läser från TCP-anslutningen, bearbetar datan till passande format och uppdaterar lokala variabler för att sedan lägga ett anrop till funktionen `UpdateCanvas()` inom kön *mainThreadQueue*. Anropen extraheras från kön genom den kontinuerligt körande funktionen `Update()`. Om kön inte är tom, utförs det inlagda anropet till `UpdateCanvas()`, vilket uppdaterar ett `Render Texture` objekt benämnt (`stereographic`), med den senaste mottagna bilden.

I detta fall är bilden som skrivs över till `Render Texture` objektet en stereografisk bild, det vill säga två delvis överlappande bilder som tillsammans möjliggör djupseende. Bilden genereras genom en kombination av Peppers två ögonkameror, vilka illustreras tydligare i Figur 3.12 och 3.13. `stereographic` är kopplad till ett `Material` objekt benämnt `stereographicMaterial`, som använder en av Unitys inbyggda *shaders*, *Skybox/Panoramic*, för att kombinera de två stereografiska bilddelarna. Materialet appliceras därefter på insidan av det tidigare nämnda spelobjektet `Sphere`, som är underordnat till `Main Camera`. Resultatet är en omslutande visuell miljö med djupseende, där sfären roterar och följer spelavatarens rörelse inom programmet.

4.2.3 TCPSend

Modulen `TCPSend` innehåller all logik för att hantera rörelsekommandon som skickas till den externa datorn kopplad till Pepper. Detta omfattar allt från insamling av data från VR-utrustningens sensorer till att bearbeta, beräkna och skicka vidare informationen. På grund av dess komplexitet är modulen uppdelad i ett flertal olika delmoduler som alla hanterar varsin funktion. Huvudmodulen som binder ihop allt är MonoBehaviour-skriptet *TCPSend.cs*, som är kopplat i Unitys hierarki till spelobjektet med samma namn, se Figur 4.8. Med spelobjektet `TCPSend` markerat, kan man i Unitys inspector se olika möjligheter att konfigurera skriptets beteende. Följande funktioner går att konfigurera:

- **No setup** - När detta är markerat kommer programmet använda standardposition för axlarna, samt den armlängd som väljs i konfigurationen istället för de värden som kommer från den setup som beskrevs i underavsnitt 4.2.1.

- **Rotation, Shoulder & Arm Debug** - Dessa inställningar aktiverar debugläge för motsvarande moduler. Rotation debug skriver ut kroppens rotationer i konsolen, shoulder debug skriver ut axlarnas position och arm debug visar de värden för armarna som räknas ut genom invers kinematik.

- **Angle Threshold** - Ställer in vid vilken tröskel som nya vinklar skickas för huvudets rotation, i radianer. Ett angle threshold på 0.1 innebär exempelvis att det endast skickas ett nytt kommando för att röra huvudet om skillnaden mot det senast skickade kommandot är större än 0.1 radianer.

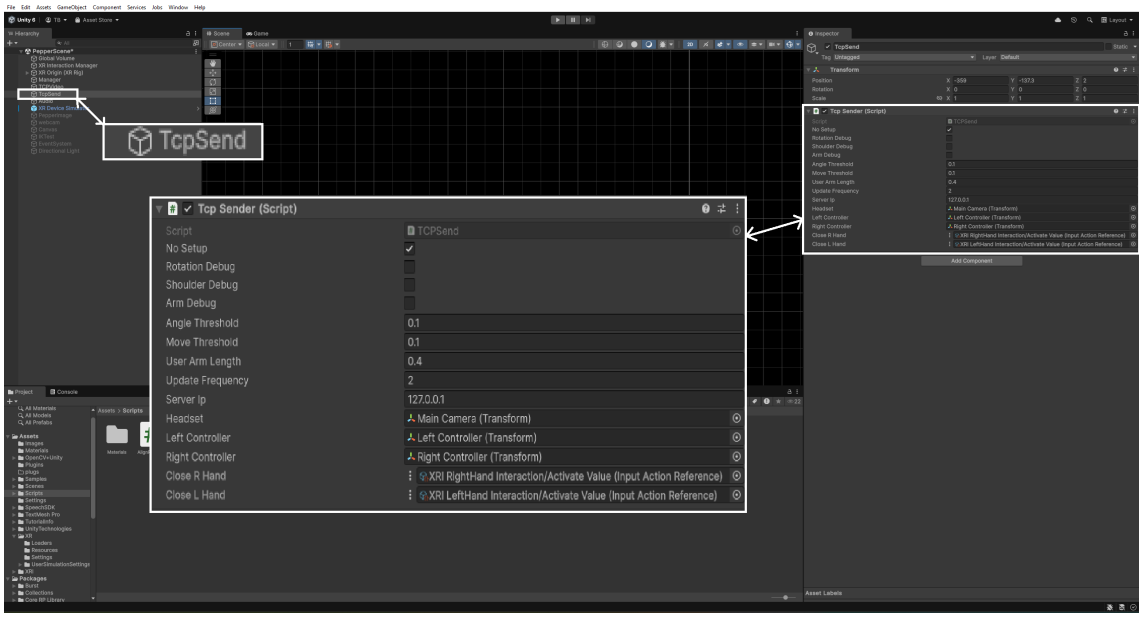
- **Move Threshold** - Likt angle threshold ställer denna in vid vilken tröskel som nya kommandon för rörelse i rummet skickas, i meter.

- **User Arm Length** - Om programmet körs utan setup ställer man in operatörens armlängd här.

- **Update Frequency** - Här kan man ställa in hur många gånger i sekunden som programmet ska göra beräkningar för robotens ledvinklar och rörelse. Om datorn som används har tillräckligt hög processorkraft kan frekvensen höjas för att minska latensen.

- **Server IP** - IP-adress för datorn som kontrollerar Pepper. Om båda program körs på samma dator används 127.0.0.1.

- **Länkning av transformer** - Resterande fält hanterar länkning av headsetets och kontrollernas position och rotation, samt kontrollernas knappar, till skriptet.



Figur 4.8: Spelobjektet TcpSend i hierarkin och dess konfigurationer.

Som tidigare nämnt är skriptet *TCPSend.cs* den modul som binder samman samtliga moduler som hanterar rörelseberäkningar till Unity-programmet. Eftersom skriptets basklass `TcpSender` ärver från `MonoBehaviour` finns det några funktioner som anropas automatiskt av Unitys runtime vid körning. I `TcpSender` används funktionerna `Start()`, `Update()` och `OnApplicationQuit()`.

`Start()`-funktionen körs en gång när skriptet aktiveras och hanterar all initialisering som krävs innan resterande delar av programmet kan köras. Detta omfattar bland annat initialisering av variabler som krävs vid start. Om programmet körs med setup hämtas axeloffset-variablerna `initialLeftShoulderPos` och `initialRightShoulderPos`, samt operatörens armlängd (`armLength`) från Manager-modulen. Om setup inte används sätts axlarnas offset till ett standardvärde på 15cm nedåt och 15cm åt sidan. `armLength` konfigureras i Unitys inspector-fönster, se Figur 4.8. Övriga variabler som initialiseras är bland annat `firstHead` och `firstShoulderRotation` som används för att lagra den initiala rotationsriktningen för headsetet när skriptet aktiveras. Ett objekt instansieras även för varje del i rörelsesystemet: armrörelse, huvudrotation och förflyttning i rummet. Varje objekts funktionalitet och ansvarsområden beskrivs mer ingående i efterföljande avsnitt. Slutligen startas två bakgrundstrådar: en som ansvarar för att upprätta en TCP-anslutning till den externa datorn och en som kontinuerligt serialiserar och skickar instruktionsdata.

Upprättning av TCP-länken sker via funktionen `ConnectToServer()`. Tråden som hanterar detta kommer att starta en TCP-klient och kontinuerligt, med en sekunds mellanrum, skicka en anslutningsförfrågan till datorn kopplad till Pepper. När anslutningen är upprättad sätts flaggan `isConnected` till *true*, vilket tillåter de övriga funktionerna i skriptet att fortsätta.

Så fort `isConnected`-flaggan är satt börjar den andra tråden kontinuerligt iterera genom funktionen `SendData()`. `SendData()` hämtar först data från `commandQueue`, vilket är ett objekt av typen `ConcurrentQueue` som finns i `System.Collections.Concurrent`. Kön bygger på en FIFO-princip (first in, first out) och är anpassad för parallell användning mellan flera trådar. Kön innehåller objekt vars statiska typ är `VRData`, vilket är en abstrakt klass som enbart innehåller ett fält: *type*. Den fungerar som basklass för mer specialiserade datatyper som innehåller information som är viktig för varje kommando: `MoveData`, `RightArmData`, `LeftArmData`, `HeadData` samt `QuitData`. Varje objekt som hämtas ur kön kodas till JSON-format och skickas sedan vidare till den externa datorn via den upprättade TCP-länken.

`Update()`-funktionen anropas från Unitys runtime en gång per frame och har i `TCPSend` två funktioner. Den ena funktionen är att uppdatera axlarnas position baserat på rotation och förflyttning i rummet. Det genomförs genom att kontinuerligt flytta och rotera vektorerna som motsvarar axlarnas offset från headsetet utifrån hur headsetets position och rotation ändras. Den andra funktionen är att kontinuerligt skicka ny, relevant data till de tre delarna i rörelsesystemet genom respektive objekts `UpdateData()`-funktion. En mer detaljerad beskrivning av vilken data som överförs till varje modul presenteras i senare avsnitt.

4.2.4 Armrörelser

För att beräkna armvinklar baserat på handkontrollernas position används skriptet *moveArm.cs*. Under `Start()`-funktionen i `TCPSend` skapas en instans av typen MoveArm som initialiseras med nödvändiga lokala variabler. Bland annat kopplas `commandQueue` från `TCPSend` för att möjliggöra överföring av data samt `closeRHand` och `closeLHand` så att knappen som kontrollerar om händerna är öppna eller stängda kan läsas av. Användarens konfiguration såsom uppdateringsfrekvens, precision, max antal iterationer och operatörens armlängd sparas även som lokala variabler. Slutligen startas en bakgrundstråd som kör funktionen `UpdateArmsLoop()`, vars uppgift är att kalla på huvudfunktionen `UpdateArms()` med jämna mellanrum bestämda av uppdateringsfrekvensen. För varje frame anropas även funktionen `UpdateData()` av Unitys huvudtråd, vars uppgift är att uppdatera de lokala variablerna för axelposition, kroppens initiala rotation, kontrollerposition samt headsetets aktuella rotation. Detta görs för att säkerställa trådsäkerhet och undvika race conditions.

`UpdateArms()` samlar ihop den data som krävs för samtliga armrörelser. Först anropas `ReadValue()` för knapparna kopplade till `rHandClosed` och `lHandClosed`, och sparas som floats. Sedan anropas `CalculateJointAngles()` för både vänster och höger arm, där beräkningar som krävs för att få ut ledvinklar till Peppers armar genomförs. En vektor som pekar från respektive axelposition till motsvarande hand konstrueras och roteras så att z-axeln pekar rakt framåt i förhållande till robotens kropp, i enlighet med Unitys konvention för koordinatsystem. Vektorns

magnitud skalas med en faktor $\frac{0.4}{\texttt{userArmLength}}$ för att passa Peppers armlängd. Därefter hanteras ett antal specialfall där operatörens hand är i ett läge som inte Pepper kan nå. Bland annat skalas vektorer som är för nära kroppen upp till det minsta avstånd från axeln som Pepper klarar av att hålla sina armar. Slutligen konverteras koordinataxlarna för att matcha Peppers koordinatsystem enligt följande regler:

Tabell 4.1: Omvandling av koordinataxlar till Peppers koordinatsystem.

Från	**Till**
x	$-y$
y	z
z	x

Den modifierade vektorn skickas in till funktionen `CalculateJointAngles()` i skriptet *InverseKinematics.cs* tillsammans med konfigurationsvariablerna samt de senaste ledvinklarna. *InverseKinematics.cs* innehåller en implementation i C# för den iterativa modellen för invers kinematik som beskrevs i underavsnitt 2.5.2 och 2.5.3. Den framåtriktade kinematiken samt beräkning av jacobimatrisen som används i den inversa kinematiken finns implementerad i skriptet *ForwardKinematics.cs* och bygger på metoderna som beskrivs i underavsnitt 2.5.1 och 2.5.2. Om ingen lösning hittas för den aktuella handpositionen returneras istället de föregående vinklarna. Slutligen sammanställs all data i ett LeftArmData- och ett RightArmData-objekt, vilka därefter skickas vidare till TCPSend via `commandQueue`.

4.2.5 Rörelse i rummet

Förflyttning i rummet hanteras på Unity-sidan av skriptet *MoveBody.cs*. Likt *MoveArm* skapas en instans av typen `MoveBody` som initieras med bland annat startposition, uppdateringsfrekvens och tröskel för uppdatering av rörelsekoordinater. Dessutom kopieras `commandQuene` från TCPSend för överföring av rörelsekommandon. En bakgrundstråd skapas för att köra funktionen `UpdateMoveLoop()` som kontinuerligt anropar funktionen `UpdateMove()` med jämna mellanrum bestämda av uppdateringsfrekvensen, likt implementationen för `MoveArm`. Funktionen `UpdateData()` uppdaterar de lokala variablerna med positionsdata från huvudtråden för att garantera trådsäkerhet.

Skriptets huvudsakliga funktionalitet sker i funktionen `UpdateMove()`. Positionsdata från VR-headsetet representeras som en tredimensionell vektor och roteras så att z-axeln alltid pekar i samma riktning som användaren hade vid uppstart. Därefter jämförs den aktuella positionen med den senast skickade. Om avståndet mellan dem överstiger den valda tröskeln, läggs ett rörelsekommando med de nya koordinaterna till i `commandQueue`.

4.2.6 Huvudrotation

I syfte att skicka applicerbar rotationsdata använder applikationen sig av skriptet *MoveHead.cs*. Vid instansiering skapas lokala variabler för initiala rotations- och po-

sitionsvärden, samt en bakgrundstråd som anropar funktionen `UpdateHeadLoop()`. Under tiden applikationen körs gör `UpdateHeadLoop()` kontinuerliga anrop till funktionen `UpdateHead()`, med en justerbar uppdateringsfrekvens, *_updateFrequency*.

`UpdateHead()` ansvarar för att bearbeta huvudets rotationsdata. Funktionen börjar med att hämta aktuell rotation från `Main Camera` i form av *Euler angles*, som lagras i *Quaternion* objektet, *_headsetRotation*. Av de tre rotationskomponenterna används x till *pitch* och y till *yaw*, medan z inte har någon motsvarande rotationsaxel för Pepper.

För att rotationsvinklarna ska ligga inom ett applicerbart intervall ($\pm 180°$), subtraheras 360° vid behov. Vektorn *headAngles* uppdateras med differensen mellan aktuell rotation och den initiala kroppsrotationen för *pitch*, samt aktuell rotation och den nuvarande kroppsrotationen för *yaw*. Detta eftersom *yaw* bör representera vridning relativt nuvarande kroppscentrum, medan *pitch* alltid bör jämföras med den initiala lutningen och är oberoende av kroppens rotation. Vinklarna konverteras därefter till radianer. *headAngles* innehåller därmed förändring i *pitch* och *yaw* relativt överkroppen, och beroende på storleken *yaw* utförs en av två operationer:

- Om *headAngles.yaw* överskrider $45°(\pi/4$ radianer), antas operatören ha vridit hela kroppen snarare än enbart huvudet. I detta fall kallas funktionen *UpdateRotation*, vilken justerar den interna variabeln för totalrotation, *_lastRotation*, och uppdaterar ett *MoveData*-objekt som skickas till Pepper.
- Så länge *headAngles.yaw* inte överskrider 45°, men ändå har förändrats mer än den angivna tröskeln (*angleThreshold*), uppdateras ett *HeadData*-objekt med aktuell *pitch* och *yaw*, vilket skickas till Pepper.

För att undvika onödiga utskick lagras senaste skickade rotationsvärden i vektorn `_lastHead`, vilken uppdateras genom en kallelse på det stödjande skriptet *Helper.cs* och dess funktion `UpdateAngles()`. Rollen av `TCPsend` i detta sker genom anrop till funktionen `UpdateData()`, som uppdaterar de lokala variablerna `_headsetRotation` och `_lastPosition` samt ger tillbaka vektorn `lastRotation`.

4.2.7 Audio

Spelobjektet `Audio`, som använder skriptet *receiveSound.cs*, skapar vid programmets start en `AudioSource` komponent. Denna modul spelar upp ljud genom att kontinuerligt uppdatera ett lokalt `AudioClip` objekt med korta ljudsegment, vilka spelas upp via Unitys inbyggda `Play()` funktion. För att undvika att blockera huvudtråden extraheras ljuddata från UDP-anslutningen i en separat tråd, `receiveAudio`. Den inkommande datan lagras som en float-array i en `ConcurrentQueue`, där `AudioSource` försöker hämta nya ljudsegment med `TryDequeue()`.

4.3 Pepper - Dator 2

Pythonprogrammet som styr Pepper med hjälp av NAOqi:s API består av flera egenskapade python-moduler:

arm.py	*head.py*	*pepper.py*
com.py	*motions.py*	*pepperaudiofeed.py*
config.py	*movement.py*	*video.py*

När NAOqi nämns i detta avsnitt är följande metod eller klass tagen från NAOqi API [49].

4.3.1 Pythonmoduler

Huvudprogrammet som körs på Dator 2 är benämnt *pepper.py*, och fungerar som en mellanhand för kommunikationen mellan Unity och Pepper. Programmet startar med att initiera datorn som en användare av Pepper, vilket sker med hjälp av *config.py*, där portnummer och IP-adresser ställs in. Därefter startas två separata processer som kör modulerna *video.py* och *pepperaudiofeed.py*. I processen som kör *video.py* initieras TCP-uppkopplingen till Dator 1 med hjälp av modulen *com.py*. Eftersom *pepperaudiofeed.py* skickar ljudet med UDP har kommunikationen implementerats direkt i modulen. Modulen *motions.py* startar fyra processer och tillhörande pipelines, en för varje arm (*arm.py*), huvudet (*head.py*) och rörelse i rummet (*movement.py*). Slutligen öppnas en ytterligare TCP-länk med programmet *TCPSend.cs* med hjälp av *com.py*, där instruktioner tas emot och sedan omdirigeras till korresponderande modul i *motions.py*.

Två klasser är definierade i *arm.py*, `LArm` och `RArm`. Skillnaden mellan klasserna ligger i benämningen på leder och begränsningsintervall, då viss data är inverterad i jämförelse. Båda klasserna kör varsin loop som tar in uppdaterade värden för korresponderande arm från Dator 1, för att sedan skickas direkt till Pepper genom att använda funktionerna `setAngles()`, `openHand()` och `closeHand()`. Processen som kör *head.py* hanterar all huvudrörelse från användaren och uppdaterar Pepper på samma sätt som *arm.py*, med hjälp av funktionen `setAngles()`.

Modulen *movement.py* som hanterar Peppers förflyttning och positionering i rummet möjliggör rörelse i alla riktningar samt fri rotation i båda riktningar. För att rotation och förflyttning ska speglas korrekt i relation med operatören har enklare logik implementerats i form av funktionerna `fix_rotation()` och `calculate_coordinates_error()`. Eftersom NAOqi:s `move()`-metod som förflyttar roboten utgår från Peppers egna kroppsriktning, samtidigt som rotationen utgår från Peppers globala koordinatsystem, används även dessa funktioner för att ta den globala rotationen i hänsyn vid förflyttning.

Videoflödet från Pepper hämtas med modulen *video.py* där funktionen `get_frame()`

hämtar en bild i taget med funktionen `getImageRemote()`, vilket inkluderas i NAOqi:s `ALVideoDevice` klass. Bilden formateras sedan till RGB-format och skickas via uppkopplad TCP-länk som en JPEG.

Modulen *pepperaudiofeed.py* hämtar ljudet från Peppers mikrofoner genom att initiera en NAOqi applikation. Efter att en applikation skapas så initieras klassen `PepperAudioFeed` med applikationen som argument, där den startas och kopplas till NAOqi klassen `ALAudioDevice`. Klassen innehåller funktionen `startProcessing()`, som bestämmer vilken samplingsfrekvens ljudet kommer ha och vilken eller vilka mikrofoner som ska användas. Den innehåller också `processRemote()` som tar in fyra argument, mängd kanaler, antal värden per kanal, tid och ljudbuffert. Två trådar används för överföring av data, en för anslutningen mellan Pepper och Dator 2, och en annan tråd för anslutningen mellan Dator 2 och Dator 1.

Filen *com.py* är modulen som upprättar TCP-länken med Dator 1. Det finns två olika klasser `ComSender` som skickar Peppers bild och `ComReceiver` som tar emot instruktioner från Dator 1.

4.3.2 Styrning

Det finns fem olika typer av meddelanden som kan skickas från Dator 1, vilka är presenterade i Figur 4.2. Positionsdatan som skickas i meddelanden från Dator 1 hämtas från TCP-länken in i modulen *pepper.py*, där första kontrollen sker. Om meddelandet är 'quit' innebär det att Unity inte längre körs och programmet på Dator 2 stängs av. Är det något av de andra fyra meddelanden kommer detta att skickas vidare till modulen *motions.py*. Där läggs datan till i rätt kö beroende på vilket meddelande som skickades: `LArm`, `RArm`, `Head` eller `Move`. Efter att datan lagts i kön plockas den upp i motsvarande modul: *arm.py*, *head.py* eller *movement.py*, som använder sig av inbyggda metoder för att röra roboten. Hela flödet visualiseras i Figur C.1.

Hur de olika modulerna är uppbyggda skiljer sig åt. *arm.py* och *head.py* arbetar med samma princip, de tar in värden på vinklar som man vill flytta roboten till och flyttar sedan med relativt låga konstanta hastigheter.

Modulen *movement.py* som styr roboten runt på golvet fungerar lite annorlunda där en regulator är implementerad. Detta gör det möjligt att kunna styra hastigheten baserat på hur mycket fel roboten är. Regulatorn använder återkoppling, vilket gör att positionsdata även måste läsas från roboten. Metoden `getRobotPosition()` skickar tillbaka robotens nuvarande x-, y-koordinat samt rotation θ. Med denna information kan robotens positionsfel användas för att räkna ut hastigheter i x- och y-riktning, samt en rotationshastighet. Därefter kan metoden `move()` som tar in dessa tre hastigheter som argument, förflytta roboten.

5
Användarstudie

En användarstudie utfördes om hur styrsystemet upplevs, både från operatörens och den interagerande användarens sida. Deltagarna i studien fick besvara enkäter innan och efter tester av systemet, som sedan analyserades. Nedan följer avsnitt om deltagare, procedur, enkäter, begränsningar och analys i användarstudien.

5.1 Deltagare

Användarstudien delades upp i två grupper där den första gruppen bestod av användare (N=11, medelålder=23.73, SD=8.62, 36% män) som fick interagera med Pepper, och den andra bestod av operatörer (N=10, medelålder=24.20, SD=4.94, 90% män) som fick styra roboten.

Majoriteten av deltagarna var studenter på högskolenivå som rekryterades på universitetsområdet. De hade mycket erfarenhet av teknologi generellt, men mindre tidigare erfarenhet av robotar. Gruppen operatörer upplevde sig dock ha något mer erfarenhet av teknologi generellt och med robotar än vad gruppen användare hade.

5.2 Procedur

De två grupperna testades separat vid olika tidpunkter, en person i taget, vilket innebar att det vid varje tillfälle kom att vara minst en person från kandidatgruppen som antingen var användare eller operatör.

Studien utfärdades i beaktande av principerna för WoZ. Operatören fick styra Pepper som om det vore dess egna kropp utefter användarens instruktioner. Pepper stod avskilt från operatören i ett annat rum, vilket medförde att användaren endast hade fysisk kontakt med Pepper, och inte med operatören. Användaren kunde ge instruktioner till Pepper via mikrofon, men operatören kunde endast svara genom kroppsliga rörelser.

5.2.1 Användare

Studien på användare som interagerade med Pepper gjordes i en lokal, avskilt från operatören, där de först fick se Pepper i ett sovande läge. Här fick användaren en introduktion om att de under testet hade fria tyglar att prata med Pepper och ge den instruktioner. Användarna var i det här läget inte medvetna om att

Figur 5.1: Exempelbild på operatör.

Figur 5.2: Exempelbild på användare.

roboten var styrd av en människa. Tanken var att användaren skulle få interagera fritt med Pepper, där endast råd på saker man potentiellt skulle kunna instruera Pepper att göra togs upp innan testets början. När användaren var redo att påbörja interaktionen startades systemet upp där en i kandidatgruppen stod som operatör i ett annat rum. Målet för operatören var att försöka följa användarens instruktioner. Några instruktioner som användare gav under testets gång var:

- Kan du gå till plats X?
- Kan du kolla/peka på objekt X?
- Kan du snurra ett varv?
- Kan du visa dina biceps?
- Nicka/skaka på huvudet för att svara ja/nej på fråga X.

5.2.2 Operatör

För testerna som utfördes på operatören fick deltagaren först en kort uppvisning på hur systemet fungerade genom att en i kandidatgruppen styrde Pepper i ett och samma rum. Efter detta placerades Pepper i ett annat rum för att simulera ett WoZ-scenario. Deltagaren fick då själv styra Pepper, men där de skulle följa en i kandidatgruppens instruktioner som stod som användare. Instruktionerna till operatören var uniforma och bestod av ett flertal steg. Den första instruktionen var att Pepper skulle nicka om den hörde användaren. Efterföljande instruktioner var sedan att Pepper skulle skaka på huvudet, snurra runt ett varv, gå till en stol några meter bort, och därefter gå till användaren. Avslutningsvis uppmanades Pepper att försöka slå till en PET-flaska som hängde från taket. Dessa steg var utformade i syfte att få operatören att uppleva varierad funktionalitet av systemet, och att alla som testade systemet skulle få en liknande upplevelse.

5.3 Enkäter

Varje deltagare besvarade en enkät både innan och efter att de hade testat systemet. Det fanns totalt fyra enkäter: Användare-innan, användare-efter, operatör-innan och operatör-efter.

Den första delen av båda gruppernas enkäter bestod av introduktionsfrågor (se Tabell B.1) och var till för att samla in demografisk data om deltagarna. Fråga 3 och 4 var på en likertskala 1-5.

Innan systemet startades upp och användaren fick interagera med Pepper fick deltagaren först svara på NARS (se avsnitt 2.6). I enkäten var en fråga "att manövrera en robot framför andra människor skulle göra mig nervös" bortplockad. Motiveringen till detta var att NARS besvarades innan interaktionen med Pepper och gjordes för att minska risken att en användare skulle misstänka Pepper som fjärrstyrd. När testet var slutfört fick användaren också svara på GODSPEED (se avsnitt 2.7). Till den sistnämnda enkäten tillkom en fråga om användaren trodde att roboten var autonom eller fjärrstyrd. När alla frågor var besvarade avslöjades det att Pepper var teleopererad av en i kandidatgruppen.

Frågorna som operatörer fick besvara innan och efter styrning av Pepper var utformade på en likertskala, 1–5, där 1 står för "Stämmer definitivt inte" och 5 står för "Stämmer helt". En 3:a som svar innebär en neutral ståndpunkt i frågan. Enkäten innan styrning handlade om deltagarnas tidigare erfarenheter, etik kring användningen av styrsystem för robotar, och förväntningar på systemet (se Tabell B.4). Enkäten efter styrning handlade om hur de upplevde styrsystemet (se Tabell B.5).

5.4 Begränsningar

Procedurens upplägg utgick från antagandet att en person som genomgick studien kunde tänkas påverkas av hur den andra personen agerade, och därmed ge skilda resultat beroende på person. Till exempel hade en användare som fått interagera med Pepper vars operatör saknade teknikvana kunnat tänkas agera annorlunda jämfört med om operatören hade haft god teknikvana. På motsvarande sätt kunde en operatör tänkas uppleva studien annorlunda beroende på vem som var användare. Exempelvis hade en användare som ger operatören instruktioner utanför styrsystemets gränser skapat en upplevelse annorlunda gentemot en användare som håller sig fåordig. Med detta i åtanke avsåg vi att hålla studien så uniform som möjligt. Operatörer fick endast instruktioner från en i kandidatgruppen, och användare interagerade endast med Pepper som var styrd av en i kandidatgruppen.

5.5 Analys

IBM SPSS 30 användes för att analysera svaren. Med SPSS kunde enkätdatan kategoriseras i variabler för att dynamiskt skapa olika typer av analyser. För att få fram ett överskådligt resultat har främst deskriptiv statistik utförts, där parametrar av typen medelvärde och standardavvikelse varit av intresse. SPSS har också använts för att undersöka den interna validiteten på de enkäter som användaren fick besvara. Detta gjordes med hjälp av Cronbach's alfa.

6
Resultat

En presentation av kandidatarbetets resultat visas i detta avsnitt. Resultatet består dels av det styrsystem som skapats, och dels av data från användarstudien.

6.1 Produkt

Baserat på tester av styrsystemet, svar från operatörer efter genomförd studie, samt de uppnådda funktionerna, kan ett samlat resultat av styrsystemets funktionalitet uppskattas. Resultatet utgår från den slutliga versionen av produkten som användes under användarstudien. Avsnittet är uppdelat i de komponenter som var och en är centrala för arbetet.

6.1.1 Rörelse

När det gäller operatörens förflyttning i rummet efterliknade Pepper med relativt god noggrannhet operatörens position. Förflyttning kunde ske i alla riktningar oberoende av kroppsrotation. Rotation av roboten följde operatören och det avstånd Pepper förflyttade sig motsvarade i stor del operatörens förflyttning. Ur ett funktionellt perspektiv uppfyllde styrsystemet sitt syfte vad gäller rörelse i rummet.

Vissa förbättringsområden kvarstår dock. En mindre fördröjning kan observeras, vilket delvis beror på systemets latens, men även acceleration som krävs av Peppers manövreringsplattform. I vissa situationer utgjorde Peppers inbyggda säkerhetssystem också ett hinder. När objekt eller personer kom för nära roboten hindrade dess sensorer fortsatt manövrering. Detta innebar att robotens rörelse i vissa fall avbröts tills den manuellt flyttades bort från hindret.

Implementationen för Peppers huvudrörelse uppfyllde i stora drag den förväntade funktionaliteten. På grund av begränsningar i antalet sensorer kopplades dock även Peppers kroppsrotation till operatörens huvudrotation. Den valda lösningen av att robotens kropp roterar när operatörens huvud vrids mer än 45° fungerade i de flesta fall som tänkt, men resulterade ibland i oavsiktlig rotation, särskilt vid snabba huvudrörelser.

Peppers armrörelser följde operatörens med varierande noggrannhet. Precisionen förbättrades avsevärt när operatörens armar höll sig inom ett rörelseområde som låg på säkert avstånd från armarnas rotationsbegränsningar. När en eller

flera rotationspunkter närmade sig dessa gränser uppstod däremot i vissa fall ett oförutsägbart rörelsemönster, vilket påverkade funktionaliteten negativt.

6.1.2 Video

Trots att kravet för visuell återkoppling uppfylls, är detta ett av områdena med störst förbättringspotential. För att möjliggöra djupseende i systemet valdes de två kamerorna i Peppers ögon som kan producera stereoskopiska bilder. Detta medförde att både upplösning och uppdateringsfrekvens försämrades aningen. I den slutliga lösningen är videon som syns i VR av relativt låg upplösning med ca 2-5 FPS. Vidare är synfältet begränsat i relation till en människas, vilket ledde till att operatörer upplevde det svårt att se sina armar och händer. En skärmbild av operatörens vy under styrningen av Pepper kan ses i Figur 6.1.

Figur 6.1: Skärmbild av operatörens vy i VR.

6.1.3 Ljud

Ljudkvaliteten som hörs från VR-utrustningen ansågs vara brusig, vilket gjorde att det inte alltid gick att höra användaren som interagerade med Pepper. Mikrofonerna, som var placerade på huvudet, befann sig dessutom i närheten av en nedkylande fläkt, vilket bidrog till ett konstant bakgrundsljud. Det brusiga ljudet i kombination med fläkten tvingade användare att prata högt, tydligt och relativt nära Pepper för att det skulle höras väl.

6.1.4 Latens

Flera operatörer kommenterade att styrningen upplevdes svår i början, då systemets latens försämrade inlevelsen i upplevelsen. I kombination med ett smalt synfält och kameror av lägre kvalitet uppfattades styrningen inte särskilt naturlig. Latensen innebar att operatörerna behövde röra sig i ett långsammare tempo för att roboten skulle kunna följa dem med högre precision.

6.2 Användarstudie

Det här avsnittet presenterar resultatet av de svar som användare och operatörer gav i samband med användarstudien. Användaren besvarade NARS (se Tabell B.2)

innan interaktionen med Pepper och GODSPEED (se Tabell B.3) efter interaktionen. Operatören svarade på frågor (se Tabell B.4) kopplade till styrsystemet både innan och efter styrningen (se Tabell B.5), men också på frågor om förväntningar och upplevelsen. Båda grupperna besvarade dessutom introduktionsfrågor innan de testade systemet (se Tabell B.1).

6.2.1 Användares svar innan interaktion

I Tabell 6.1 nedan presenteras det sammanställda resultatet på introduktionsfrågorna (exklusive kön och ålder) för användarna (N=11, medelålder=23.73, SD=8.62, 36% män). IF3 och IF4 hade ett medelvärde på 3.55 (SD=1.29) respektive 1.91 (SD=1.14). Resultatet visar att användarna hade högre generell erfarenhet av teknologi än erfarenhet av robotar. Spridningen på svaren var stor.

Tabell 6.1: Deskriptiv statistik från introduktionsfrågorna.

Fråga	Medelvärde	SD
IF3	3.55	1.29
IF4	1.91	1.14

I Tabell 6.2 nedan presenteras statistiken från enkätdatan som användare svarade på innan interaktionen med Pepper. För totalen blev medelvärdet 3.36 (SD=0.54), vilket tyder på att användarna i genomsnitt svarade att deras generella känslor för robotar var något negativa, men nära neutrala. Den interna validiteten var acceptabel (Cronbach's alfa=0.74). Vidare kan man se att användarna i genomsnitt hade en rädsla för robotars potentiella påverkan på samhälle och kultur, samt skepsis mot att utveckla emotionella band till robotar, där medelvärdet för S2 var 4.05 (SD=0.54) och S3 3.60 (SD=0.88). Slutligen ses i statistiken att S1 (negativa känslor inför att interagera med robotar) var den enda delskalan som visade sig ha en viss lutning för det positiva på genomsnittsanvändaren, där medelvärdet var 2.53 (SD=0.64).

Tabell 6.2: Deskriptiv statistik och intern validitet för NARS och dess delskalor.

Skala	Medelvärde	SD
Totalen	3.36	0.54
S1	2.53	0.64
S2	4.05	0.54
S3	3.60	0.88

6.2.2 Användares svar efter interaktion

I Tabell 6.3 nedan presenteras statistiken från enkätdatan som användare svarade på efter interaktionen med Pepper. Resultatet som visas tyder på att den genomsnittliga användaren tycktes ha ett neutralt intryck av Pepper i sin helhet. Totalen hade ett medelvärde på 2.97 (SD=0.57) och en hög intern validitet (Cronbach's alfa=0.90). För de enskilda skalorna från GODSPEED varierade resultatet. Den genomsnittliga användaren svarade över 3 på S3, S4 och S5, med ett medelvärde på 3.51 (SD=0.78),

Tabell 6.3: Deskriptiv statistik och intern validitet för GODSPEED och dess delskalor.

Skala	Medelvärde	SD
Totalen	2.97	0.57
S1	2.04	0.63
S2	2.67	0.62
S3	3.51	0.78
S4	3.58	0.70
S5	3.36	0.61

3.58 (SD=0.70) respektive 3.36 (SD=0.61). Detta kan tolkas som att användare i genomsnitt uppfattade Pepper mer åt det intelligenta, trevliga, och säkra hållet.

På S1 och S2 svarade användare i genomsnitt 2.04 respektive 2.67. Att medelvärdet låg på 2.04 för S1 pekar på att Pepper inte uppfattades som människolik, utan mer maskinlik. Ett medelvärde på 2.67 för S2 antyder att Pepper uppfattades något mer livlös än livfull.

Det sista användare fick svara på efter GODSPEED var "Upplevde du roboten som autonom eller fjärrstyrd?", varav 10 svarade autonom och 1 svarade fjärrstyrd. Detta kan tolkas som att alla förutom en deltagare inte misstänkte att Pepper var teleopererad.

6.2.3 Operatörers svar innan styrning

I Tabell 6.4 nedan presenteras det sammanställda resultatet på introduktionsfrågorna (exklusive kön och ålder) för operatörerna (N=10, medelålder=24.20, SD=4.94, 90% män). IF3 och IF4 hade ett medelvärde på 4.10 (SD=0.74) och 2.30 (SD=1.34) vardera. Detta kan tolkas som att operatörerna hade mer tidigare erfarenhet av teknologi generellt jämfört med de som interagerade med Pepper. Däremot hade de mindre erfarenhet av specifikt robotar.

Tabell 6.4: Deskriptiv statistik från introduktionsfrågorna.

Fråga	Medelvärde	SD
IF3	4.10	0.74
IF4	2.30	1.34

Ett resultat har tagits fram på de frågor (se Tabell B.4) som operatören besvarade innan styrning av Pepper. I tabell Tabell 6.5 nedan presenteras statistiken från enkätdatan.

På de första två frågorna som berörde tidigare erfarenheter om VR och åksjuka, så svarade operatörer i genomsnitt att de varken hade mycket erfarenhet av VR eller lätt för att uppleva åksjuka. OI1 hade ett medelvärde på 2.40 (SD=1.35) och

OI2 hade ett medelvärde på 2.50 (SD=1.78). Den höga standardavvikelsen för dessa frågor tyder på att deltagarnas svar varierade stort mellan individer.

Tabell 6.5: Deskriptiv statistik på enkäten som operatörer svarade på innan styrning.

Fråga	Medelvärde	SD
OI1	2.40	1.35
OI2	2.50	1.78
OI3	4.60	0.52
OI4	3.90	0.99
OI5	4.40	0.70
OI6	4.00	1.16
OI7	4.50	0.71

OI3, OI4 och OI5 handlade om operatörers känslor och etiska tankar kring styrsystemet. Här svarade operatörer i genomsnitt högt. På OI3, om de kände sig bekväma med att de skulle styra en robot, svarade operatörer i genomsnitt 4.60 (SD = 0.52), vilket indikerar att deltagarna kände sig väldigt bekväma med att styra en robot. På OI4, om de kände sig bekväma med att en användare inte har vetskap om att hen styr roboten, var medelvärdet på 3.90 (SD=0.99), vilket tyder på att operatörerna kände sig bekväma med att en användare inte har vetskap om att roboten är fjärrstyrd. På OI5, som handlade om de skulle känna sig bekväma med att användaren fick reda på att de styrt roboten i efterhand, svarade operatörer 4.40 (SD=0.70) i genomsnitt. Detta visar att deltagarna kände sig väldigt bekväma med att avslöja att de styrt roboten under experimentet.

De sista två frågorna, OI6 och OI7, handlar om operatörernas förväntningar på robotens rörelseförmåga och styrsystem. Resultaten visar att operatörerna i genomsnitt hade höga förväntningar. På frågan om de förväntade sig att deras nya robotkropp skulle kunna plocka upp föremål från marken var medelvärdet 4.00 (SD=1.16), vilket tyder på en hög förväntan på systemet. På frågan om de förväntade sig att sin nya robotkropp skulle efterlikna deras rörelser var medelvärdet 4.50 (SD=0.71), vilket indikerar att de hade väldigt höga förväntningar på styrsystemets latens- och rörlighetsförmåga.

6.2.4 Operatörers svar efter styrning

Ett resultat har tagits fram på de frågor (se Tabell B.5) som operatören besvarade efter styrning av Pepper. Statistiken från denna data kan läsas i Tabell 6.6 nedan. Den allmänna upplevelsen av hur bra det gick att styra roboten tycktes vara neutral, där operatörer i genomsnitt svarade 3.10 (SD=1.37) på OE1. Den höga standardavvikelsen indikerar en stor variation i deltagarnas svar.
Operatörer tyckte att Pepper följde deras rörelser till en högre nivå än neutralt där medelvärdet på OE2 var 3.40 (SD=1.17). Dock upplevdes en högre grad av latens hos flera av operatörerna där medelvärdet på OE3 var 3.70 (SD=0.68).

Tabell 6.6: Deskriptiv statistik på enkäten som operatörer svarade på efter styrning.

Fråga	Medelvärde	SD
OE1	3.10	1.37
OE2	3.40	1.17
OE3	3.70	0.68
OE4	1.80	0.79
OE5	2.60	0.84
OE6	1.80	1.03
OE7	2.00	0.94

Videokvalitén upplevdes problematisk där medelvärdet på OE4 var 1.80 (SD=0.79). Vidare fick OE5, om det kändes som man hade djupseende, ett något lägre värde än neutralt med ett medelvärde på 2.60 (SD=0.84). Det var få som kände en stark känsla av åksjuka, då operatörer i genomsnitt svarade 1.80 (SD=1.03) på OE6.

Slutligen svarade operatörer 2.00 i genomsnitt (SD=0.94) på OE7. Detta tyder på att de inte fick känslan av att deras medvetande existerade inom roboten.

7
Diskussion

Detta avsnitt diskuterar centrala delar av den utvecklade systemarkitekturen, samt resultaten från användarstudien. Syftet är att evaluera hur arbetet tacklar de uppställda frågeställningarna, och identifiera möjliga förbättringsområden.

7.1 Styrsystem

Följande avsnitt diskuterar centrala tekniska områden relaterade till systemarkitekturen. Berörda avsnitt omfattar rörelse, bild, samt visuell- och auditiv återkoppling till operatören. Diskussionen baseras utifrån användarfeedback och uppnådd teknisk funktion.

7.1.1 Visuell- och auditiv återkoppling

I syfte att konstruera en immersiv upplevelse för operatören vid styrningen av Pepper var den visuella och auditiva återkopplingen till operatören ett av projektets centrala områden. Operatörens synfält bestod konsekvent av en stereografisk lösning som möjliggjorde djupseende likt en människas. Lösningen upplevdes däremot som otillräcklig av flera operatörer, där upplösningen, bildfrekvensen och synfältets bredd lyftes som ett återkommande problem. Kombinationen av dessa tre faktorer bidrog till att operatörerna hade svårt att avgöra hur väl Pepper följde deras rörelser. Även när roboten faktiskt speglade förflyttningarna väl, var det svårt för operatören att tolka detta visuellt. Det mest berörda området var armarna, eftersom kamerans begränsade periferi innebar att de inte gick att se under större delen av användningen. Då människor i hög grad förlitar sig på visuell återkoppling, antas detta möjligtvis vara den största negativa faktorn i utvärderingen under studien.

Om man bortser från möjligheten att förbättra robotens hårdvara, finns det ett antal potentiella förbättringsåtgärder för att adressera problemet. För att minska operatörens osäkerhet kring hur väl roboten speglar rörelserna, föreslås att en liten visualisering av hela robotmodellen integreras i ett hörn av operatörens VR-miljö, exempelvis via NAOqi:s Choregraphe-miljö. Detta skulle kunna ge operatören en mer direkt förståelse för robotens position och rörelser i realtid. En ytterligare möjlig förbättring vore att implementera en funktion för att växla mellan kamerorna direkt via VR-kontrollerna.

Även den auditiva återkopplingen visade sig vara bristande. Ljudet från ro-

boten var av låg kvalitet, vilket gjorde det svårt för operatören att uppfatta instruktioner. En potentiell orsak till detta var mikrofonens placering nära en inbyggd fläkt, vilket tros vara anledningen till ett konstant svagt klickande ljud och förvrängt läte. För att förbättra den auditiva återkopplingen bör mikrofonplacering och ljudöverföring optimeras.

7.1.2 Rörelse inom rummet

Observationer från studien visade att förflyttningen av Pepper inom rummet upplevdes som relativt intuitiv och fungerande. Operatörer hade nästintill inga problem med att navigera roboten framåt, bakåt och i sidled med god precision. Den största förbättringspotentialen låg dock i hanteringen av kroppens rotation.

I det nuvarande systemet roterar Peppers kropp endast vid rotation av huvudet som överstiger 45°. Denna lösning är tänkt att endast köras under helkroppsrotation, men orsakade i vissa fall ofrivilliga rotationer. I scenarion där operatören ville fokusera blicken på ett objekt mot periferin av kameran, feltolkades ibland rörelsen av systemet som en avsiktlig rotation, vilket orsakade förvirring.

En möjlig förbättring för framtida implementationer vore att separera behandlingen av huvudets och kroppens rotation med hjälp av en extern sensor på bröstkorgen. Peppers rotation skulle därav baseras på bröstkorgens riktning, medan robotens huvud följer VR-headsetets rotation. På så vis kan operatören fritt rikta sin blick utan att riskera oönskad kroppsrörelse hos roboten.

En regulator utvecklades även för att stabilisera rörelser i rummet i syfte för ytterligare stabilisering. Efter tester med regulatorn på armarna märktes det dock att systemet blev betydligt långsammare och att latensen ökade markant. Regulatorn valdes därför bort för specifikt armarna, och styrs istället med en låg konstant hastighet. Denna låga hastighet valdes eftersom roboten upplevdes som ryckig vid högre hastigheter.

7.1.3 Styrning av armar

Att konstruera en metod för smidig och exakt styrning av robotens armar var ett av de mest utmanande momenten i projektet. Ursprungligen användes en enkel geometrisk metod där trigonometriska satser, såsom cosinussatsen, utnyttjades för att approximativt beräkna armarnas vinklar. På grund av komplexa beräkningar för att korrekt anpassa vinklarna för shoulder pitch och shoulder roll utifrån armbågens riktning hamnade armen däremot ibland i fel position. Dessutom blev armarna mer begränsade i sina rörelser, specifikt när armarna rörde sig framför robotens kropp eller i sidled. Utöver det hade den geometriska lösningen även buggar som orsakade armrörelser som inte alls motsvarade operatörens rörelser. På grund av dessa begränsningar valde vi istället att implementera en iterativ version av invers kinematik, vilket möjliggjorde en mer exakt positionering av armarna.

Invers kinematik gav bättre resultat, men det var fortfarande svårt att få armrörelserna helt bra. Framför allt är det viktigt att armarna befinner sig inom ett område som Pepper kan nå för att det ska vara möjligt att hitta en lösning som kan verifieras med framåtriktad kinematik. För att undvika problem orsakade av skillnad i längd mellan användarens och Peppers arm implementerade vi därför en skalningsfaktor för att få armen inom det tillåtna intervallet. Trots detta finns det fortfarande ett flertal gränsfall som roboten inte kunde nå, vilket i vissa fall ledde till rörelser som inte matchar operatörens rörelser.

En annan begränsning som upptäckts är att den axelposition och armlängd som beräknas i setup-programmet ofta ger sämre resultat för styrningen än att använda fördefinierade standardvärden. Det har framför allt märkts när den fördefinierade armlängden varit kortare än den som beräknats vid setup, vilket ger ett intryck av att styrningen känns bättre om rörelserna sker inom ett kortare intervall. Detta är ett område som kan utvecklas vidare med bättre metoder för att översätta operatörens armrörelser till rörelser möjliga för Pepper.

Ett annat potentiellt forskningsområde är hur olika implementationer av invers kinematik påverkar exempelvis robotens rörelser och latens. Med en utökad modell för invers kinematik som, utöver position, även tar hänsyn till orientering, är det möjligt att armrörelserna blir mer exakta, men det finns även en risk för högre latens på grund av ökade beräkningar.

7.1.4 Transportprotokoll

Det finns flera anledningar till varför man skulle vilja använda sig av TCP för styrningen av Pepper. Den främsta av dessa anledningar är att Pepper är väldigt begränsad i dess rörelser, vilket innebär att det är kritiskt att programmet planerar rörelserna för roboten korrekt. Armrörelser som mottas i fel ordning höjer risken att roboten inte hittar rätt rörelsebana, alltså blir TCP:s garanti för ordningsföljd betydelsefullt för hur väl roboten följer operatörens rörelser.

En annan fördel med TCP är säkerheten. TCP kräver en etablerad anslutning mellan sändare och mottagare, vilket skyddar mot oönskad trafik. UDP, som saknar denna mekanism, tillåter mottagning av paket från vilken avsändare som helst. Detta leder till en ökad risk för att främmande sändare skickar störande eller skadliga datapaket som kan påverka robotens beteende eller krascha programmet.

Om man bortser från säkerhetsaspekten med TCP så tycker vi att UDP hade varit bättre för att skicka ljud och bild. Faktum är att det enda vi använder UDP för är att skicka ljudet mellan datorerna. De SDK:er som finns tillgängliga för roboten använder sig av TCP vilket är den största anledningen bakom varför vi själva använder TCP. Att föra över mer funktionalitet till UDP hade minskat latensen på programmet, dock hade skillnaden i latens för ljud och bild inte påverkats så mycket.

Det borde noteras att vi inte har testat att styra roboten genom UDP. En

minskad latens kanske inte uppfattas vid kortare tester, men kan säkert vara betydande för operatörens upplevelse under längre användning. En operatör som blir åksjuk kommer inte att märka av de negativa effekterna direkt, utan det byggs upp över tid. En minskad latens kan fördröja åksjuka för operatörer, och kanske hindrar det helt för vissa. Vi vet inte heller i vilken grad programmet hade haft svårare att hitta rörelsebanor.

Ett intressant område att forska på är därför till vilken grad en UDP-anslutning påverkar upplevelsen. Det är fullt möjligt att den minskade latensen hade minskat åksjuka och förbättrat inlevelsen då roboten följer operatörens rörelser mer exakt. Det är också rimligt att anta att en försämring av rörelsebanornas beräkning försämrar operatörens upplevelse. Vad som lämpar sig bäst för styrsystemet är svårt att säga utan ytterligare tester.

7.2 Användarstudie

Innan diskussion kring användarstudiens resultat sker är det viktigt att nämna det begränsade antalet deltagare som var med i studien. Det relativt låga antalet på 11 personer för användare och 10 personer för operatörer gör att resultatet inte är lika tillförlitligt som om antalet deltagare hade varit av en större mängd. Trots det begränsade antalet deltagare som besvarat formulären bedöms det som tillräckligt för att kunna föra en diskussion om de angivna forskningsfrågorna.

7.2.1 Hur upplever operatörer styrsystemet?

Operatörer hade i genomsnitt höga förväntningar på styrsystemet. På frågan om de trodde att roboten skulle kunna plocka upp något från marken svarade de i genomsnitt 4.00. Denna fråga var tänkt att ge ett svar på hur rörlig operatören förväntade sig att deras nya robotkropp skulle vara. Vidare förväntade sig operatörerna att roboten skulle efterlikna deras rörelser med ett medelvärde på 4.50. Efter det att operatören hade upplevt systemet svarade de att Pepper följde deras rörelser till en högre nivå än neutralt, med ett medelvärde på 3.40, men som var lägre än förväntningarna.

Att operatörernas upplevelse av systemet inte motsvarade deras förväntningar kan bero på flera faktorer. Det är naturligt att ha höga förväntningar på en teknisk produkt om inga tydliga förvarningar givits om dess begränsningar. När det gäller styrsystemet berodde upplevelsen sannolikt på en kombination av systemets latens och videoflöde. Latensen innebar en fördröjning innan roboten följde operatörens rörelser, vilket kunde få särskilt ovana användare att uppleva en bristande koppling mellan egna rörelser och Peppers. Detta stöds av frågan “Kändes det som det tog lång tid innan roboten rörde sig som din riktiga kropp?”, som hade ett medelvärde på 3.70. Videoflödet kan också ha påverkat användarnas upplevelse, då det utgjorde den primära källan till återkoppling för operatören. Ju svårare det är att se om en rörelse utförs, desto större osäkerhet är det kring om roboten verkligen följer rörelsen. Den upplevda låga videokvalitén, med ett medelvärde

på 1.80, kan därför ha bidragit till att styrningen inte kändes tillfredsställande. Sammanfattningsvis tyder resultaten på att ett system med lägre latens och bättre videokvalité sannolikt hade lett till en mer positiv användarupplevelse, och är därmed värt att prioritera i vidare utveckling.

7.2.2 Hur upplevs interaktionen med roboten?

Resultatet som gavs av svaren på GODSPEED visar på att den generella uppfattningen av Pepper var neutral där medelvärdet för totalen var 2.97. Om man jämför detta med vad användarna svarade på NARS innan interaktionen så var också deras generella känslor inför robotar relativt neutrala med ett medelvärde på 3.36. Ett antagande skulle kunna göras att användarnas tidigare känslor gentemot robotar skulle kunna ha en påverkan på hur de upplevde interaktionen, och i sådana fall förklara det relativt neutrala resultatet på båda enkäterna. Med detta i åtanke hade det varit intressant att i framtida studier se om resultatet från NARS har en korrelation med resultatet från GODSPEED, då det inte räcker att göra denna slutsats från endast denna studie.

Från delskalorna av GODSPEED sticker framförallt S1 ut, där medelvärdet på 2.04 tyder på att Pepper uppfattades som maskinlik. En förklaring till detta skulle kunna vara en kombination av Peppers uppenbara robotiska utseende och att rörelserna stundvis kunde se mekaniska ut jämfört med en människa. Vidare visar resultatet från GODSPEED:s delskalor att Pepper upplevdes vara något trevlig, intelligent, och att användarna kände sig något säkra kring Pepper. S4, den upplevda intelligensen hos Pepper, hade ett medelvärde på 3.58. Det var flera användare som under testets gång reagerade chockartat på Peppers intelligensnivå och undrade hur ett så avancerat AI-system var byggt.

På den allra sista frågan, om användaren trodde att Pepper var autonom eller fjärrstyrd, svarade 10 av 11 deltagare att de trodde den var autonom. Detta stämmer överens med de reaktioner som deltagarna hade på interaktionen under testets gång. Det var ingen som verbalt kommenterade om Pepper faktiskt var styrd. Tvärtom så handlade de uttalade tankarna exempelvis om hur det var möjligt att skapa ett autonomt system till denna grad, eller frågor om vilka instruktioner som Pepper förstod sig på. Efter att den sista frågan var besvarad avslöjades det att Pepper var fjärrstyrd, där majoriteten av deltagarna blev överraskade.

7.2.3 Övriga observationer

Efter det att operatörerna hade testat systemet svarade de i genomsnitt att de upplevde en låg grad av åksjuka, vilket är gynnsamt då VR-system tenderar att skapa åksjuka. Att de knappt upplevde någon åksjuka kan dels bero på att de var individer som i allmänhet inte har lätt för att uppleva åksjuka, vilket stämmer överens med vad de svarade innan styrning. Framför allt kan det dock tänkas bero på att testet endast varade i ett par minuter, vilket kan ha varit en för kort tid för att hinna uppleva åksjuka. I framtida studier kan det därför vara av intresse att

låta operatörerna testa systemet under en längre period.

De deltagare som var med i användarstudien plockades till största del upp från universitetsområdet. Med detta i åtanke skulle ett antagande kunna göras att de flesta var studenter, vilket i sin tur skulle kunna förklara medelsnittsåldern på 23.73 år hos användarna och 24.20 år hos operatörerna. Ytterligare, hade den unga åldern kombinerat med deras troligtvis tekniska inriktning kunnat förklara det relativt höga genomsnittssvaret på introduktionsfrågan “Hur mycket erfarenhet har du med teknologi generellt?”, där användare svarade 3.55 och operatörer 4.10.

I framtida studier kan det vara intressant att undersöka resultat när både användaren och operatören, från var sin sida, är nya till systemet och genomgår studien samtidigt. Detta var däremot inte av intresse i denna studie då det hade behövts betydligt fler personer att testa på, vilket på grund av den givna tidsramen behövde begränsas till två isolerade grupper.

7.3 Samhälleliga och etiska aspekter

I samband med att teknologin inom social robotik utvecklas, kan teleoperation av humanoida robotar möjligen få en större roll i framtidens samhälle. Robotar kan fungera som representanter i miljöer där fysisk närvaro inte är möjlig eller praktisk. Det kan omfatta personer med sjukdomar, rörelsenedsättningar, psykisk ohälsa eller de som regelbundet behöver resa längre sträckor. Enligt oss har teknologin potential att öka delaktighet i både arbetsliv och sociala sammanhang.

Denna utveckling kan dock medföra utmaningar. Om fjärrstyrning av robotar normaliseras som alternativ till fysisk närvaro, finns risken att mänsklig interaktion ersätts snarare än kompletteras. I värsta fall kan detta påverka hur relationer byggs, och leda till en framtid där tekniken istället distanserar människor. Det finns även risk att personer som hade kunnat delta fysiskt väljer tekniska lösningar av bekvämlighet, vilket kan påverka både personlig utveckling samt social upplevelse.

I framtida tillämpningar kan också integritet och tillit bli centrala frågor. Om det inte tydligt framgår när en människa styr en robot, kan det skapa förvirring och obehag. Otydlig kommunikation kan få människor att känna sig lurade eller exponerade, vilket i sin tur kan påverka hur tekniken uppfattas och i vilken grad den accepteras.

Vi anser att framtidens användning av dessa system kräver tydliga etiska riktlinjer. För att främja ansvarsfull användning bör det vara transparent när det är en människa bakom styrningen samtidigt som tekniken inte bör ersätta sociala miljöer i större grad. Om robotar får en större roll i samhället blir det viktigt att införandet sker varsamt och med respekt för samhällets existerande strukturer.

8

Slutsats

Projektet har resulterat i ett fullt fungerande system där operatören kan kontrollera Peppers position i rummet, huvudrörelser och armrörelser med godtagbar latens och precision. Operatören får dessutom återkoppling i form av ljud och bild i realtid från Peppers perspektiv, vilket möjliggör en mer naturlig interaktion med användaren. Systemets begränsningar är främst relaterade till hårdvara, där kameraupplösning, bildhastighet, bristande sensorer hos operatören samt Peppers begränsade rörelseomfång påverkar upplevelsen av systemet negativt.

Den första forskningsfrågan var "Hur kan man implementera ett VR-baserat styrsystem för Pepper?". Detta projekt har visat att det kan implementeras genom att använda en spelmotor med stöd för VR. Genom att vidarebefordra Peppers kameraflöde och omgivningsljud till operatörens VR-utrustning, samtidigt som VR-utrustningens positionsdata översätts till rörelsedata som Pepper kan tolka, skapas ett system där operatören kan kontrollera Pepper ur ett förstapersonsperspektiv.

Gällande den andra forskningsfrågan, "Hur upplever operatörer styrsystemet?", följde roboten i stort sett operatörernas rörelser väl, men upplevelsen levde inte helt upp till deltagarnas förväntningar. Områdena i störst behov av vidareutveckling var latens, bildkvalitet och bildfrekvens.

För den tredje forskningsfrågan, "Hur upplevs interaktionen med roboten?", framgick det att interaktionen med Pepper upplevdes varken positivt eller negativt. Pepper uppfattades som maskinlik, trevlig, intelligent och säker. 10 av 11 användare upplevde Pepper som autonom vilket visar att systemet lyckades med att simulera en interaktion mellan människa och autonom robot.

Vidareutveckling av projektet rekommenderas fokusera på att förbättra återkopplingen från Pepper till operatören, minska systemlatens samt integrera fler sensorer för en mer realistisk styrning. Förbättringar i mjukvara och effektivisering av systemets responstid kan öka realismen i rörelserna och därmed förbättra upplevelsen för både operatör och användare.

Projektets kod och program finns på GitHub: `https://github.com/Kandidatarbete-27/vr-styrsystem-pepper`.

Litteratur och referenser

[1] J. Du m.fl, "Global Automation Humanoid Robot: The AI accelerant," Goldman Sachs, New York, NY, USA, 2024. [Online]. Tillgänglig: https://www.goldmansachs.com/pdfs/insights/pages/gs-research/global-automation-humanoid-robot-the-ai-accelerant/report.pdf, Hämtad: 2025-05-13.

[2] S. Mukherjee, M. M. Baral, S. K. Pal, V. Chittipaka, R. Roy och K. Alam, "Humanoid Robot in Healthcare: A Systematic Review and Future Research Directions," i *Proceedings of the 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, Faridabad, Indien, 2022, ss. 822–826. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/9850577, Hämtad: 2025-05-13.

[3] M. Tzampazaki, E. Vrochidou och G. A. Papakostas, "Social Robots in Education: To Select or Not to Select a Robot for a Teaching Subject at an Educational Level?," i *2024 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Kroatien, 2024, ss. 1–6. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/10721629, Hämtad: 2025-05-13.

[4] R. Y. Kim, "Anthropomorphism and Human-Robot Interaction," *Commun. ACM*, vol. 67, nr. 2, ss. 80–85, Feb. 2024, doi:10.1145/3624716.

[5] R. Vinjamuri, "Preface," i *Discovering the Frontiers of Human-Robot Interaction: Insights and Innovations in Collaboration, Communication, and Control*, R. Vinjamuri, Red., Cham, Schweiz: Springer Cham, 2024, ss. vii–viii. Tillgänglig: https://link.springer.com/book/10.1007/978-3-031-66656-8, Hämtad: 2025-05-13.

[6] D. Kragic och Y. Sandamirskaya, "Effective and natural human-robot interaction requires multidisciplinary research," *Sci. Robot*, vol. 6, nr. 58, eabl7022, Sep. 2021. [Online]. Tillgänglig: https://www.science.org/doi/10.1126/scirobotics.abl7022, Hämtad: 2025-05-13.

[7] M. A. Goodrich och A. C. Schultz, "Human–Robot Interaction: A Survey," *Foundations and Trends® in Human–Computer Interaction*, vol. 1, nr. 3, ss. 203–275, Jan. 2008, doi:10.1561/1100000005.

[8] J. Nasir, P. Oppliger, B. Bruno och P. Dillenbourg, "Questioning Wizard of Oz: Effects of Revealing the Wizard behind the Robot," i *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, Neapel, Italien, 2022, ss. 1385–1392. [Online]. Tillgänglig: https://ieeexplore.ieee.org/document/9900718, Hämtad: 2025-05-14.

[9] F. Rietz, A. Sutherland, S. Bensch, S. Wermter och T. Hellström, "WoZ4U: An Open-Source Wizard-of-Oz Interface for Easy, Efficient and Robust HRI Experiments," *Frontiers in robotics and AI*, vol. 8, 668057, Jul. 2021, 10.3389/frobt.2021.668057.

[10] M. Gamboa, S. Thunberg, P. Alves-Oliveira och M. B. Loerakker, "We Are the Robots: Tapping Into the Lived Experiences of Wizards of Oz," i *CHI EA '25: Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, Yokohoma, Japan, 2025, ss. 1–8. [Online]. Tillgänglig: `https://dl.acm.org/doi/10.1145/3706599.3720149`, Hämtad: 2025-05-19.

[11] S. Thellman, J. Lundberg, M. Arvola och T. Ziemke, "What Is It Like to Be a Bot?: Toward More Immediate Wizard-of-Oz Control in Social Human-Robot Interaction," i *HAI '17: Proceedings of the 5th International Conference on Human Agent Interaction*, Bielefeld, Tyskland, 2017, ss. 435–438. [Online]. Tillgänglig: `https://dl.acm.org/doi/abs/10.1145/3125739.3132580`, Hämtad: 2025-05-14.

[12] Softbank Robotics. "Meet Pepper: The Robot Built for People," [Online]. Tillgänglig: `https://us.softbankrobotics.com/pepper` (hämtad: 2025-05-15).

[13] A. M. Oliveira, I. B. Q. de Araujo, "A Brief Overview of Teleoperation and Its Applications," i *2023 15th IEEE International Conference on Industry Applications (INDUSCON)*, São Bernardo do Campo, Brasilien, 2023, ss. 601–602. [Online]. Tillgänglig: `https://ieeexplore.ieee.org/document/10374937`, Hämtad: 2025-05-14.

[14] T. B. Sheridan, "Human–Robot Interaction: Status and Challenges," *Human Factors*, vol. 58, nr. 4, ss. 525–532, Jun. 2016, doi:10.1177/0018720816644364.

[15] D. McColl, Z. Zhang och G. Nejat, "Human Body Pose Interpretation and Classification for Social Human-Robot Interaction," *Int J of Soc Robotics*, vol. 3, ss. 313–332, Aug. 2011, doi:10.1007/s12369-011-0099-6.

[16] A. Niculescu, B. van Dijk, A. Nijholt, H. Li och S. L. See, "Making Social Robots More Attractive: The Effects of Voice Pitch, Humor and Empathy," *Int J of Soc Robotics*, vol. 5, ss. 171–191, Jan. 2013, doi:10.1007/s12369-012-0171-x.

[17] H. Kiilavuori, V. Sariola, M. J. Peltola och J. K. Hietanen, "Making eye contact with a robot: Psychophysiological responses to eye contact with a human and with a humanoid robot," *Biological Psychology*, vol. 158, 107989, Jan. 2021, doi:10.1016/j.biopsycho.2020.107989.

[18] V. D. Corte, F. Sepe, D. Gursoy och A. Prisco, "Role of trust in customer attitude and behaviour formation towards social service robots," *International Journal of Hospitality Management*, vol. 114, 103587, Sep. 2023, doi:10.1016/j.ijhm.2023.103587.

[19] D. Fischinger m.fl, "Hobbit, a care robot supporting independent living at home: First prototype and lessons learned," *Robotics and Autonomous Systems*, vol. 75, ss. 60–78, Jan. 2016, doi:10.1016/j.robot.2014.09.029.

[20] J. Sellergren, "Roboten Hobbit – ett trevligt sällskap," *Lunds universitet*, [Online], Okt. 13, 2016. Tillgänglig: `https://www.lu.se/artikel/roboten-hobbit-ett-trevligt-sallskap` (hämtad: 2025-05-14).

[21] M. F. El-Muhammady, S. A. M. Zabidi, H. M. Yusof, M. A. Rashidan, S. N. Sidek och A. S. Ghazali, "Initial Response in HRI: A Pilot Study on Autism Spectrum Disorder Children Interacting with a Humanoid QTrobot," i *Robot Intelligence Technology and Applications 7*, Daejeon, Korea, 2022, ss. 393–406. [Online]. Tillgänglig: `https://link.springer.com/chapter/10.1007/978-3-031-26889-2_36`, Hämtad: 2025-05-14.

[22] LuxAI, "QTrobot, Educational robot for at home education of children with autism," 2021. [Online]. Tillgänglig: `https://luxai.com/assistive-techrobot-for-special-needs-education` (hämtad: 2025-05-14).

[23] Aldebaran, "Aldebaran - Pepper," 2018. [Online]. Tillgänglig: https://aldebaran.com/en/pepper (hämtad: 2025-05-14).

[24] A. Henschel, G. Laban och E. S. Cross, "What Makes a Robot Social? A Review of Social Robots from Science Fiction to a Home or Hospital Near You," *Curr Robot Rep*, vol. 2, ss. 9–19, Mar. 2021, doi:10.1007/s43154-020-00035-0.

[25] Aldebaran, "Technical overview - Aldebaran 2.5.11.14a documentation," 2018. [Online]. Tillgänglig: `http://doc.aldebaran.com/2-5/family/pepper_technical/index_pep.html` (hämtad: 2025-05-14).

[26] J. F. Kelley, "An iterative design methodology for user-friendly natural language office information applications," *ACM Transactions on Information Systems*, vol. 2, nr. 1, ss. 26–41, Jan. 1984, doi:10.1145/357417.357420.

[27] A. Hamad, och B. Jia, "How Virtual Reality Technology Has Changed Our Lives: An Overview of the Current and Potential Applications and Limitations," *International journal of environmental research and public health*, vol. 19, nr. 18, ss. 1-11, Sep. 2022, doi:10.3390/ijerph191811278.

[28] TDK Corporation, "Behind the Metaverse: A Closer Look at VR Devices and their Ultra-Compact Sensors," 2023. [Online]. Tillgänglig: `https://www.tdk.com/en/featured_stories/entry_051-metaverse-motion-sensors.html` (hämtad: 2025-05-14).

[29] "Virtual Reality (VR) Market Size, Share, Growth Trends, Industry Statistics Consumer Analysis (2025 - 2030)," Mordor Intelligence, Hyderabad, Indien, 2024. [Online]. Tillgänglig: `https://www.mordorintelligence.com/industry-reports/virtual-reality-market`. Hämtad: 2025-05-14.

[30] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, nr. 4, ss. 425–432, Apr. 1980, doi:10.1109/TCOM.1980.1094702.

[31] A. S. Tanenbaum och D. J. Wetherall, *Computer Networks*. 5. uppl., Boston, MA, USA: Pearson Education, 2010.

[32] J. F. Kurose och K. W. Ross, *Computer Networking: A Top-Down Approach*. 8. uppl., Harlow, Storbrittanien: Pearson Education, 2021.

[33] N. Olifer och V. Olifer, *Computer Networks: Principles, Technologies and Protocols for Network Design*, Chichester, England: John Wiley & Sons Ltd, 2006.

[34] J. Postel, "Transmission Control Protocol," RFC 793, Sep. 1981, doi:10.17487/RFC0793.

[35] J. Postel, "User Datagram Protocol," RFC 768, Aug. 1980, doi:10.17487/RFC0768.

[36] J. Postel, “Internet Protocol,” RFC 791, Sep. 1981, doi:10.17487/RFC0791.
[37] T. Eriksson, “Kinematik,” i *Nationalencyklopedin*. [Online]. Tillgänglig: http://www.ne.se, hämtad: 2025-05-15.
[38] Y. Suga, “Python Pepper Kinematics,” *Github*, [Online]. Oct. 2015. Tillgänglig: https://github.com/sugarsweetrobotics/python_pepper_kinematics (hämtad: 2025-04-27).
[39] R. Nilsson, “Inverse Kinematics,” masteruppsats, Institution för dataveteskap och elektroteknik, Luleå tekniska universitet, Luleå, Sverige, 2009.
[40] S. R. Buss, “Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods,” Institution för matematik, University of California San Diego, San Diego, CA, USA, Oct. 2009. [Online]. Tillgänglig: https://mathweb.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf, Hämtad: 2025-05-15.
[41] J. Nocedal och S. J. Wright, *Numerical Optimization*, 2. uppl., New York, NY, USA: Springer, 2006. [Online]. Tillgänglig: https://link.springer.com/book/10.1007/978-0-387-40065-5, Hämtad: 2025-05-15.
[42] *The Damped Pseudo Inverse*. https://robotics.caltech.edu/~jwb/courses/ME115/handouts/damped.pdf. Kursmaterial från ME115, California Institute of Technology. 2010.
[43] T. Nomura, T. Suzuki, T. Kanda och K. Kato, “Altered attitudes of people toward robots: Investigation through the Negative Attitudes toward Robots Scale,” i *Proceedings of the AAAI-06 Workshop on Human Implications of Human-Robot Interaction*, Menlo Park, CA, USA, 2006, ss. 29–35. [Online]. Tillgänglig: https://www.researchgate.net/publication/228368017_Altered_attitudes_of_people_toward_robots_Investigation_through_the_Negative_Attitudes_toward_Robots_Scale, Hämtad: 2025-05-15.
[44] S. Thunberg, S. Thellman och T. Ziemke, “Don’t Judge a Book by its Cover: A Study of the Social Acceptance of NAO vs. Pepper,” i *HAI ’17: Proceedings of the 5th International Conference on Human Agent Interaction*, Bielefeld, Tyskland, 2017, ss. 443–446. [Online]. Tillgänglig: https://dl.acm.org/doi/10.1145/3125739.3132583, Hämtad: 2025-05-19.
[45] C. Bartneck, D. Kulić, E. Croft och S. Zoghbi, “Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots,” *Int J of Soc Robotics*, vol. 1, nr. 1, ss. 71–81, Jan. 2009, doi:10.1007/s12369-008-0001-3.
[46] Aldebaran, “NAOqi Framework - NAO Software 1.14.5 documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/1-14/dev/naoqi/index.html (hämtad: 2025-05-15).
[47] Aldebaran, “SDKs - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/dev/programming_index.html (hämtad: 2025-05-15).
[48] Aldebaran, “Choregraphe Suite - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/software/choregraphe/index.html (hämtad: 2025-05-15).

[49] Aldebaran, “NAOqi APIs - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/naoqi/index.html (hämtad: 2025-05-15).

[50] Aldebaran, “Joints - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html (hämtad: 2025-05-15).

[51] Aldebaran, “ALMotion - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/naoqi/motion/almotion.html (hämtad: 2025-05-15).

[52] Aldebaran, “Body frames - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/masses_pep.html (hämtad: 2025-05-15).

[53] Aldebaran, “Links - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/links_pep.html (hämtad: 2025-05-15).

[54] Aldebaran, “ALMotion API - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-4/naoqi/motion/almotion-api.html (hämtad: 2025-05-15).

[55] Aldebaran, “2D Cameras - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/video_2D_pep.html (hämtad: 2025-05-15).

[56] Aldebaran, “Stereo Camera - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/video_Stereo_pep.html (hämtad: 2025-05-15).

[57] Aldebaran, “Actuator Sensor list - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/pepper_dcm/actuator_sensor_names.html (hämtad: 2025-05-15).

[58] Aldebaran, “Inertial unit - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/inertial_pep.html (hämtad: 2025-05-15).

[59] Aldebaran, “Contact and tactile sensors - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/contact-sensors_pep.html (hämtad: 2025-05-15).

[60] Aldebaran, “Sonars - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/sonar_pep.html (hämtad: 2025-05-15).

[61] Aldebaran, “Infra-Red - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/irspot_pep.html (hämtad: 2025-05-15).

[62] Aldebaran, “Microphones - Aldebaran 2.5.11.14a documentation,” 2018. [Online]. Tillgänglig: http://doc.aldebaran.com/2-5/family/pepper_technical/microphone_pep.html (hämtad: 2025-05-15).

[63] Meta Platforms Inc., “Meta Quest 2: Tekniska specifikationer,” 2020. [Online]. Tillgänglig: https://www.meta.com/se/quest/products/quest-2/tech-specs (hämtad: 2025-04-21).

[64] A. Williams, "It Is The End Of The Road For Two Meta Quest VR Headsets," *Forbes*, [Online]. Jun. 30, 2024. Tillgänglig: https://www.forbes.com/sites/andrewwilliams/2024/06/30/it-is-the-end-of-the-road-for-two-meta-quest-vr-headsets (hämtad 2025-04-21).

[65] Meta Platforms Inc., "Learn how headset tracking works on Meta Quest | Hjälp för Quest | Meta Store," 2025. [Online]. Tillgänglig: https://www.meta.com/sv-se/help/quest/598701621088668 (hämtad: 2025-04-21).

[66] Meta Platforms Inc., "Introducing Oculus Air Link, a Wireless Way to Play PC VR Games on Oculus Quest 2, Plus Infinite Office Updates, Support for 120 Hz on Quest 2, and More," 2021. [Online]. Tillgänglig: https://www.meta.com/sv-se/blog/introducing-oculus-air-link-a-wireless-way-to-play-pc-vr-games-on-oculus-quest-2-plus-infinite-office-updates-support-for-120-hz-on-quest-2-and-more (hämtad: 2025-04-21).

[67] N. Bonfiglio, "DeepMind partners with gaming company for AI research," *The Daily Dot*, [Online]. Maj. 21, 2021. Tillgänglig: https://www.dailydot.com/debug/unity-deempind-ai (hämtad: 2025-03-30).

[68] Unity, "Unity 6 Releases & Support: LTS & Updates Releases | Unity," 2024. [Online]. Tillgänglig: https://unity.com/releases/unity-6/support (hämtad: 2025-04-02).

[69] Khronos Group, "OpenXR Overview - The Khronos Group Inc," [Online]. Tillgänglig: https://www.khronos.org/openxr (hämtad: 2025-03-30).

[70] ChatGPT (GPT-4o), OpenAI, 2024. [Large Language Model]. https://chat.openai.com.

A

Appendix 1

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & \cos(\theta) & -\sin(\theta) & y \\ 0 & \sin(\theta) & \cos(\theta) & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation kring X-axeln.

$$\begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & x \\ 0 & 1 & 0 & y \\ \sin(\theta) & 0 & \cos(\theta) & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation kring Y-axeln.

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & x \\ \sin(\theta) & \cos(\theta) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation kring Z-axeln.

Figur A.1: Transformationsmatriser för rotation kring axlar med translation.

$$T_1 = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T1: Rotation kring Y-axeln (shoulder pitch).

$$T_2 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T2: Rotation kring Z-axeln (shoulder roll).

$$T_3 = \begin{bmatrix} \cos(9^\circ) & 0 & \sin(9^\circ) & 0.18 \\ 0 & 1 & 0 & 0.015 \\ -\sin(9^\circ) & 0 & \cos(9^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T3: Överarmslängd, axeloffset och 9° fast vinkeloffset.

$$T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T4: Rotation kring X-axeln (elbow yaw).

$$T_5 = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T5: Rotation kring Z-axeln (elbow roll).

$$T_6 = \begin{bmatrix} 1 & 0 & 0 & 0.15 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T6: Underarmslängd och rotation kring X-axeln (wrist yaw).

$$T_7 = \begin{bmatrix} 1 & 0 & 0 & 0.07 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.03 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T7: Offset från handleden till ändeffektorns (handens) mittpunkt.

Figur A.2: Transformationsmatriser för varje steg i robotens framåtriktade kinematik för vänster arm (höger arm har samma struktur med inverterad axeloffset).

B

Appendix 2

Tabell B.1: ID för frågor, introduktionsfrågor (**IF**).

ID	Fråga
IF1	Kön?
IF2	Hur gammal är du?
IF3	Skulle du säga att du har mycket erfarenhet med teknologi generellt?
IF4	Skulle du säga att du har mycket erfarenhet med robotar?

Tabell B.2: ID för frågor, NARS (**NARS**).

*Inverterad fråga

ID	Fråga
NARS1	Det vore obehagligt om robotar verkligen hade känslor.
NARS2	Något dåligt skulle kunna hända om robotar utvecklades till levande varelser.
NARS3	Jag skulle känna mig bekväm med att prata med robotar.*
NARS4	Det vore obehagligt om jag fick ett jobb där jag var tvungen att använda robotar.
NARS5	Jag skulle kunna bli vän med robotar som har känslor.*
NARS6	Tanken på att vara med robotar som har känslor är betryggande.*
NARS7	Ordet ”robot” saknar betydelse för mig.
NARS8	Jag avskyr tanken på att robotar eller artificiella intelligenser skulle bedöma saker.
NARS9	Tanken på att stå framför en robot gör mig väldigt nervös.
NARS10	Om jag förlitar mig för mycket på robotar riskerar något dåligt att hända.
NARS11	Att prata med en robot skulle få mig att känna mig paranoid.
NARS12	Jag befarar att robotar skulle kunna ha dåligt inflytande på barn.
NARS13	Jag tror att det framtida samhället kommer att domineras av robotar.

Tabell B.3: ID för frågor, GODSPEED (**GS**).

ID	Motsatta ord		
MÄNNISKOLIKHET			
GS1	Onaturlig	-	Naturlig
GS2	Maskinlik	-	Människolik
GS3	Omedveten	-	Medveten
GS4	Artificiell	-	Levande
GS5	Rör sig stelt	-	Rör sig elegant
LIVFULLHET			
GS6	Död	-	Levande
GS7	Statisk	-	Livful
GS8	Mekanisk	-	Organisk
GS9	Artificiell	-	Levande
GS10	Trög	-	Interaktiv
GS11	Apatisk	-	Responsiv
TREVLIGHET			
GS12	Ogillar	-	Gillar
GS13	Ovänlig	-	Vänlig
GS14	Otrevlig	-	Trevlig
GS15	Obehaglig	-	Behaglig
GS16	Förfärlig	-	Bra
UPPLEVD INTELLIGENS			
GS17	Inkompetent	-	Kompetent
GS18	Ignorant	-	Kunnig
GS19	Ansvarslös	-	Ansvarsfull
GS20	Ointelligent	-	Intelligent
GS21	Dåraktig	-	Förnuftig
UPPLEVD SÄKERHET			
GS22	Orolig	-	Avslappnad
GS23	Agiterad	-	Lugn
GS24	Avvaktande	-	Överraskad

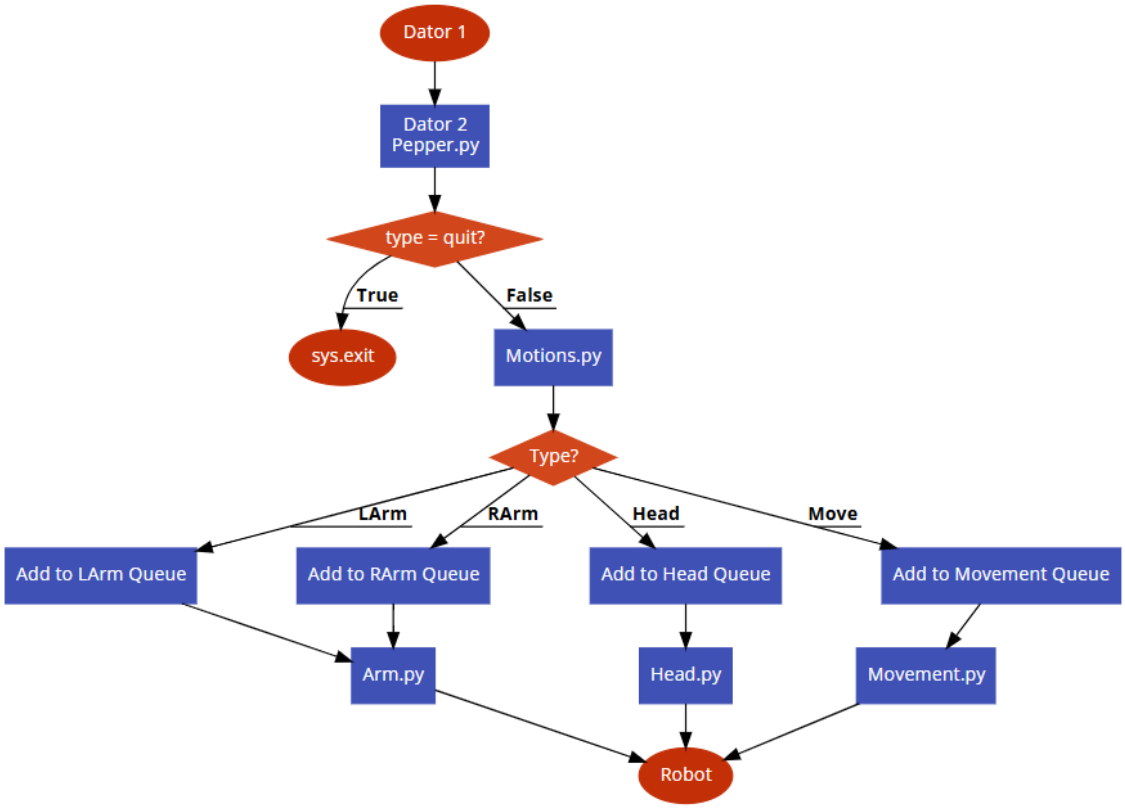
Tabell B.4: ID för frågor, operatör innan (**OI**).

ID	Fråga
OI1	Skulle du säga att du har mycket erfarenhet med virtual reality (VR)?
OI2	Skulle du säga att du har lättare för att uppleva åksjuka än andra människor?
OI3	Känner du dig bekväm med att du kommer styra en robot?
OI4	Känner du dig bekväm med att användaren inte vet om att du styr roboten?
OI5	Skulle du känna dig bekväm om användaren får reda på att du styr roboten efteråt?
OI6	Förväntar du dig att din nya robot-kropp kan plocka upp något från marken?
OI7	Förväntar du dig att din nya robot-kropp kommer efterlikna dina rörelser?

Tabell B.5: ID för frågor, operatör efter (**OE**).

ID	Fråga
OE1	Fungerade det bra att styra roboten?
OE2	Kändes det som att roboten följde dina rörelser?
OE3	Kändes det som det tog lång tid innan roboten rörde sig som din riktiga kropp?
OE4	Kändes videokvalitén bra?
OE5	Kändes det som du hade djupseende?
OE6	Upplevde du en stark känsla av åksjuka?
OE7	Kändes det som att ditt medvetande existerade inom roboten?

C
Appendix 3



Figur C.1: Dataflöde för styrning genom Python.

D
Appendix 4

D.1 Generativ AI uttalande

LLMs (Large Language Models), i detta fall ChatGPT [70], har använts under projektets gång för brainstorming, felsökning av kod och grammatikkontroll. Den har framför allt fungerat som ett stöd för att bolla idéer angående lösningar på olika problem.