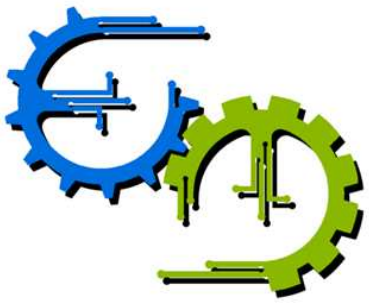
TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE SISTEMA DE TELEOPERAÇÃO PARA ROBÔ HUMANOIDE

Henrique de Sousa e Silva Balbino



UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE SISTEMA DE TELEOPERAÇÃO PARA ROBÔ HUMANOIDE

Henrique de Sousa e Silva Balbino

Trabalho de Graduação submetido como requisito parcial de obtenção do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Antônio Padilha L. Bó, ENE/UnB _____
Orientador

Prof. Mariana Costa Bernardes Matias, FGA/UnB _____
Co-Orientador

Prof. Geovany Araújo Borges, ENE/UnB _____
Examinador Externo

Dr. Roberto Souza Baptista, LARA/UnB _____
Examinador Interno

Brasília, 12 de dezembro de 2016

FICHA CATALOGRÁFICA

BALBINO, HENRIQUE DE SOUSA E SILVA
Desenvolvimento de sistema de teleoperação para robô humanoide [Distrito Federal] 2016.
xi, 49p., 210 x 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2016).
Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.

1. Teleoperação
2. NAO
3. Realidade virtual
4. HMD
I. Mecatrônica/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

BALBINO, H.S.S. (2016). Desenvolvimento de sistema de teleoperação para robô humanoide, Trabalho de Graduação em Engenharia de Controle e Automação, Publicação TG-030/2016, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 49p.

CESSÃO DE DIREITOS

Faculdade de Tecnologia - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedico este trabalho aos meus pais e às minahs irmãs por todo o carinho, apoio, atenção e educação que eles me ofereceram.

AGRADECIMENTOS

Primeiramente gostaria de agradecer aos meus orientadores Antônio Padilha e Mariana Bernardes. Antes deles se tornarem meus orientadores, eles foram os professores responsáveis pela equipe UnBeatables, que me permitiu ter o contato com o robô NAO e obter a minha experiência de programação com robótica. Além disso, graças aos conselhos da professora Mariana, fui capaz de aplicar diversos conceitos e ferramentas na execução deste trabalho.

Também gostaria de agradecer à equipe UnBeatables como um todo. Depois de participar de várias competições, ter passado muito tempo programando e testando as várias partes do código, que muitas vezes não funcionaveam, para que o robô fosse capaz de jogar futebol, fui capaz de obter muita experiência como programador e estudante de engenharia em uma aplicação prática do que aprendi durante o curso, além de ter feito novas amizades e conhecido outros lugares do Brasil e do mundo.

Em especial, gostaria de agradecer aos ex-membros da UnBeatables De Hong Jung, Rafael Lima e Marcela Pinheiro de Carvalho. Rafael várias vezes me apoiou, escutando os problemas durante a execução do trabalho e tentando oferecer soluções, além de ter me emprestado o Myo, que foi uma peça essencial para que este trabalho se tornasse completo. Marcela foi uma das grandes responsáveis execução do meu trabalho por ter desenvolvido um sistema de teleoperação com o Kinect, no qual me baseei e aproveitei diversas partes do código que me auxiliaram muito, principalmente com relação à implementação do código em ROS além do que já havia sido feito em Kinect. De Hong também foi de grande ajuda por ter me ensinado como instalar e utilizar o código da Marcela, além da implementação do giroscópio no código para o movimento da cabeça do robô. De Hong também utilizou o sistema no Hackaton da Globo, que foi uma ótima forma de testar o código em uma aplicação para o público.

Gostaria de agradecer aos meus amigos Bruno Caxito, Deivid Vale, Raphael Arthur e Jhonantans Moraes pela paciência, por terem me ajudado na resolução de alguns problemas da execução do sistema e pela ajuda em alguns testes que foram essenciais para a obtenção dos resultados obtidos.

Por fim, quero agradecer à minha família, pela atenção, carinho, suporte e educação desde sempre. Ao meu pai Luiz Antonio pelo sustento e pelo exemplo de pessoa que se esforça muito para obter os resultados que quer. À minha mãe Luciane, pelo esforço em cuidar, educar dar carinho e atenção para cada um dos seus filhos. E às minhas três irmãs Vanessa, Paula e Fernanda por toda a felicidade e diversão que passamos juntos.

RESUMO

Neste trabalho é desenvolvido uma solução para que uma pessoa possa teleoperar o robô humanoide NAO com os movimentos de seu corpo de forma que o robô seja capaz de realizar tarefas que uma pessoa também consegue fazer, incluindo andar, mover os braços e interagir com objetos no ambiente. Além disso, também é implementada uma forma do teleoperador visualizar o ambiente em que o robô se encontra na perspectiva do robô, lhe oferecendo a sensação de imersão no ambiente. O método utilizado para teleoperar o robô à distância envolve capturar vários dos movimentos e gestos que um ser humano é capaz de fazer através de múltiplos sensores e processar as informações dos sensores para transformá-los nos movimentos que o robô irá executar. Além disso, para oferecer a sensação de imersão ao usuário, como se o usuário estivesse no lugar onde está o corpo do robô, é trasmitida a imagem da câmera no robô que é visualizada no formato de realidade virtual. Dessa forma é possível simular a presença física do usuário no ambiente em que se encontra o robô.

Palavras Chave: Teleoperação, NAO, realidade virtual, telepresença, HMD.

ABSTRACT

In this work is developed a solution to which a person can teleoperate the humanoid robot NAO with the movements of his own body in a way that the robot is capable of accomplish tasks that a persoan can also do, including walking, moving the arms and interacting with objects in the environment. Beyond that, it is also implemented a way so the teleoperator can visualize the environment in which the robot is located. The method utilized to teleoperate the robot from distance involves capturing many of the movements and gestures that a human being is capable of doing through multiple sensors and process the data of the sensors to transform them into the movements that the robot will execute. Futhermore, to offer the sense of imersion to the user, just as if the user is in the place where the robot's body is, the image from the camera on the robot is transmitted to be visualized in virtual reality mode. In this way, it is possible to simulate the physical presence of the user in the environment where the robot is.

Keywords: Teleoperation, NAO, virtual reality, telepresence, HMD.

Quatérnio unitário

Símbolos

$\mathbf{q}$	Quatérnio unitário
$\underline{\mathbf{q}}$	Quatérnio dual
$\underline{a}$	Número dual
$\vec{v}$	Vetor
u, v, w	Escalares
R	Matriz de rotação
θ, ρ, φ	Ângulos

Siglas

ROS	Robot Operating System
IMU	Inertial measurement unit
HMD	Head-mounted display

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O ser humano é muito dependente de máquinas para realizar inúmeras tarefas no dia a dia. Seja para transporte, como carros e aviões, realizar algum tipo de processamento, como uma máquina de lavar, uma cafeteira, ou um micro-ondas, criar e moldar objetos como robôs industriais e impressoras 3D, entre inúmeros outros exemplos. A principal razão para emprego de máquinas nas vidas das pessoas é para torná-las mais fácil, sendo que muitas delas só é necessário que se aperte alguns botões para que realizem as tarefas para as quais foram designadas. Uma grande vantagem de se utilizar máquinas é que elas não se cansam, só precisam ser alimentadas por energia e muitas delas superam as limitações físicas dos seres humanos sendo, por exemplo, capazes de carregar cargas muito pesadas ou por longas distâncias. Por outro lado, várias dessas máquinas também exigem treinamento e conhecimentos específicos para serem manipuladas e executarem as tarefas para as quais foram designadas. Em contrapartida, os seres humanos se acostumam a utilizarem seus próprios corpos desde quando nascem, ou seja, a partir de uma certa idade todo ser humano sem nenhuma deficiência é capaz de manipular objetos com as mãos, carregar objetos, andar e executar diversas tarefas sem nenhum treinamento ou novo aprendizado.

Este trabalho é inspirado na ideia de uma pessoa poder controlar um robô humanoide com os movimentos naturais de seu corpo, os quais são reproduzidos por este robô, e de visualizar as imagens da câmera deste robô em tempo real, oferecendo a esta pessoa a sensação de que ela está imersa na realidade do robô, semelhante ao se controlar um avatar de realidade virtual. Como um robô pode possuir vantagens mecânicas em relação a um corpo humano (ser mais forte, ou mais resistente, além de ser substituível), uma possível aplicação deste trabalho seria poder efetuar tarefas que exijam mais força ou em ambientes perigosos ou de difícil acesso utilizando um corpo robótico no lugar de uma pessoa.

Com os avanços da tecnologia nas áreas da computação e da robótica, a aquisição e transmissão de dados e informações se tornou muito mais fácil e rápida. E com a manipulação dessas informações se tornou possível utilizar tais informações para controlar dispositivos mecânicos, como por exemplo os atuadores de um robô, que são capazes de realizar o que humanos nem sempre conseguem utilizando diretamente seus próprios corpos. Hoje em dia existem computadores com elevadíssimo poder de processamento, sendo comum um computador pessoal realizar até bilhões de operações por segundo. Também ocorreram avanços com sensores e atuadores, sendo mais fácil capturar informações sobre o ambiente ou algum sistema e transmitir essas informações, além do controle de partes mecânicas como um braço robótico ter se tornado mais precisa.

Graças a estes avanços tecnológicos, neste trabalho é implementada uma possível solução para que um ser humano seja capaz de controlar um corpo robótico com uma estrutura semelhante à sua (um robô humanoide), de forma que se pareça que está controlando o próprio corpo e que o usuário desta solução se sinta imerso no ambiente que o robô se encontra. Com isso, o usuário será capaz de realizar tarefas que faria normalmente usando seu próprio corpo, como se locomover no ambiente

e interagir com os objetos do ambiente ao seu redor, além de poder realizar outras tarefas que não seriam possíveis usando seu próprio corpo devido a limitações físicas se a plataforma robótica for capaz de superar tais limitações. Possíveis exemplos de aplicações para o sistema desenvolvido neste trabalho seriam: interagir com pessoas com o corpo robótico, teleoperar um robô de resgate que pode remover os escombros de um desabamento ou entrar em um local com incêndio, um ator teleoperar um robô para cenas de teatro ou de filmes, além de ser possível diversas realizar tarefas à distância.

1.2 DEFINIÇÃO DO PROBLEMA

A estrutura básica do corpo humano é composta por cabeça, membros superiores e inferiores e todos eles são ligados ao tronco. No corpo humano também estão presentes várias partes responsáveis por captar diversos tipos de informação vindas do ambiente, sejam elas de natureza luminosa, sonora, térmica, química ou mecânica. Esta estrutura permite ele seja capaz de realizar muitas tarefas incluindo se locomover no ambiente em que se encontra, perceber as características do ambiente que o cerca, interagir com e reagir ao que existe no ambiente ao seu redor para realizar algum tipo de tarefa. É necessário utilizar uma solução para captar, processar e transmitir muitos tipos de informação a respeito dos movimentos e gestos realizados pelo ser humano para que um robô humanoide consiga realizar as tarefas que um ser humano também é capaz de fazer.

Devido à estrutura que o corpo humano possui, para captar as ações e os gestos que ele é capaz de executar, são utilizados vários sensores para o sistema capazes de capturar os movimentos que cada uma dessas partes realiza. Com a captura das informações sobre os movimentos que o ser humano executa através dos sensores, é feito o processamento das informações para gerar os comandos que definem para onde as partes do robô teleoperado deverão se mover. O corpo do robô humanoide é semelhante, porém diferente, ao corpo de um ser humano, então é necessário fazer compensações para gerar os comandos de movimento do robô. Com o robô sendo controlado por um ser humano, ele deve ser capaz de executar pelo menos algumas das ações que um ser humano também e capaz, como andar no ambiente, gesticular com os braços, mover a cabeça e interagir com objetos do ambiente, como empurrá-los e agarrará-los com os dedos.

É necessário também que o usuário do sistema seja capaz de perceber o que acontece no ambiente em que o robô teleoperado se encontra. A captura das informações do ambiente em que o robô se encontra é feita por uma câmera de vídeo, cujas imagens são transmitidas para o teleoperador. O teleoperador visualiza as imagens em um dispositivo HMD acoplado à sua cabeça, de forma que as imagens são mostradas em um formato de realidade virtual. Isso permite que o usuário veja o mundo pela perspectiva do robô e lhe oferece a sensação de que o corpo do robô é o seu próprio, já que o sistema simula a presença física do teleoperador no ambiente em que se encontra o robô.

Neste trabalho, para capturar os gestos de movimentos do usuário do sistema, são utilizados vários dispositivos como Kinect, Myo e um smartphone com uma unidade inercial (IMU) embutida. Os dados obtidos por esses sensores são transmitidos e processados em um computador por um software parcialmente desenvolvido neste trabalho. O software é também responsável por utilizar os dados processados para gerar os comandos para onde irão se mover as partes do robô, para que este obedeça

aos movimentos e gestos do teleoperador e o robô execute as tarefas que um ser humano também é capaz. O software desenvolvido roda no framework chamado de Robot Operating System (ROS), que é responsável por gerenciar o processamento e a comunicação simultâneas das diversas que recebem, processam e enviam os dados utilizados para a teleoperação. A imagem da câmera acoplada ao robô também é transmitida para o computador e retransmitida para o smartphone no formato de realidade virtual. O smartphone é utilizado em conjunto com um Cardboard para que possa ser utilizado como um head-mounted display (HMD). Dessa forma, o usuário do sistema consegue visualizar as imagens da câmera para que ele se sinta imerso na realidade do robô.

Como o sistema implementado utiliza muitos módulos, um dos grandes desafios no desenvolvimento deste trabalho é, além de fazer cada parte funcionar como esperado, integrar todas elas em um único sistema e funcionando simultaneamente, além de garantir a transferência contínua das informações entre cada uma das partes, para que seja possível fornecer ao usuário a solução de controle e imersão proposta.

1.3 CONTRIBUIÇÕES DESTE TRABALHO

Com a conclusão deste trabalho, foi possível integrar diversas tecnologias de hardware e software e mostrar que com a combinação de todas elas é possível de ser criado um sistema de teleoperação que controla um robô humanoide. Também foi feita a programação para que seja capaz de transmitir, receber e processar diversos tipos de informação de vários dispositivos que são utilizados para que o controle do robô seja executado.

Boa parte deste trabalho foi aproveitada por outros trabalhos feitos anteriormente, e neste trabalho foi possível encontrar uma aplicação prática unindo todos esses outros trabalhos previamente desenvolvidos. Em especial, no trabalho de Carvalho[1], onde foi desenvolvido o controle do robô NAO por cinemática inversa utilizando o Kinect, sendo capaz de validar a utilização do controle desenvolvido nesse trabalho aplicando-o no sistema de teleoperação aqui descrito. A partir do trabalho de Carvalho, também foram desenvolvidas outras aplicações no trabalho aqui descrito, expandindo as suas aplicações para que seja possível fazer o robô caminhar a partir dos dados do Kinect e aproveitando a programação desenvolvida no trabalho de Carvalho.

1.4 TRABALHOS RELACIONADOS

Abaixo são citados alguns exemplos de outros trabalhos que utilizaram algum sistema de teleoperação em robôs humanoides.

No trabalho feito por Stanton et al.[2], é realizada a captura de todo o corpo do teleoperador através de uma roupa de captura de movimentos. É feito um treinamento com os dados coletados da roupa para transformar os dados obtidos nas posições das juntas do robô NAO. Para garantir a estabilidade, as posições das juntas do quadril e dos joelhos não foram definidas pelo treinamento, sendo substituído pelo cálculo da posição relativa do quadril e do joelho com o tornozelo, de forma

que os pés do robô sempre ficassem em paralelo com o chão.

Os resultados obtidos no trabalho de Stanton et al.[2] foram as taxas de erros medidas como uma porcentagem do alcance do movimento de cada motor. Foi obtido um erro médio de 5.55% entre todos os motores. Foi possível colocar o robô em diversas posições movendo os braços e as pernas, incluindo o robô inclinar o corpo para os lados e para frente e agachar.

Uma das grandes diferenças entre o trabalho de Stanton et al.[2] e o aqui apresentado é a forma em que são capturados os movimentos, as posições e os gestos do corpo do teleoperador para gerar as posições das juntas do robô. No trabalho de Stanton et al. é utilizado apenas uma roupa de captura, enquanto no trabalho aqui descrito são utilizados Kinect, Myo e sensores inerciais, além de não ser utilizado nenhum tipo de treinamento dos dados.

Em [3], é desenvolvido por Rodriguez et al. um sistema de teleoperação com Kinect e ROS para controlar os braços do NAO além do uso de reconhecimento de gestos e de fala para tornar a teleoperação mais humanizada. Os gestos são utilizados neste sistema para que o NAO obedeça aos comandos: ficar de pé, agachar, parar, mover para frente ou para trás, mover para a esquerda ou para a direita, girar para a esquerda ou para a direita. O reconhecimento de fala recebe a fala do teleoperador, que é processada para que o robô possa reproduzir as palavras ou executar algum dos seguintes comandos: ficar de pé, sentar, mover para frente ou para trás, mover para a esquerda ou para a direta, girar para a esquerda ou para a direita.

Foram propostos alguns experimentos no trabalho de Rodriguez et al.[3] com vários participantes para testar o sistema. O primeiro experimento ocorre com o robô sendo teleoperado por um circuito com obstáculos, sendo que o teleoperador deve ser capaz de levar o robô do início até o objetivo final. No segundo experimento, o robô deve pegar o objeto que está sendo segurado por uma pessoa e carregá-lo até outra pessoa que irá coletar o objeto. Todos os participantes foram capazes de compeltar ambos os experimentos, mas a maioria deles necessitaram de várias tentativas até serem capaz de completarem o objetivo do experimento.

Existem várias semelhanças na detecção pelo Kinect para a execução dos movimentos e dos gestos desenvolvidos no sistema desenvolvido por Rodriguez et al.[3] e no sistema aqui descrito. Porém, no trabalho de Rodriguez et al. não é utilizada a parte de imersão para que o teleoperador tenha um feedback mais próximo da realidade do robô, mas no trabalho aqui descrito também não é feito nenhum tipo de reconhecimento de fala.

Goza et al. [4] descrevem um sistema de controle por telepresença para o robô Robonaut. Como é possível perceber pelo seu nome, o Robonaut é um robô humanoide desenvolvido para ser usado no espaço e que possui 45 graus de liberdade. Este robô é operado à distância por um teleoperador que está imerso na realidade do robô. Além disso, o robô deve ser capaz de realizar tarefas manuais que exigem um alto nível de destreza, assim como os humanos são capazes de realizar com as suas próprias mãos. São utilizados sensores nas mãos, no peito e na cabeça do teleoperador para mover os braços e a cabeça do robô. Também é utilizado um par de luvas para realizar o controle dos dedos. Além disso, o teleoperador utiliza um equipamento de realidade virtual para poder estar imerso na realidade do robô.

Como o Robonaut é controlado utilizando os mesmos movimentos que um ser humano, foi con-

cluído que teleoperar o Robonaut é uma tarefa fácil e intuitiva. Além disso, mesmo sendo teleoperado à longas distâncias (entre cidades), o tempo de resposta do sistema era em torno de 0,5 segundo, que não foram suficientes para que o teleoperador sentisse alguma dificuldade em utilizar o robô.

Uma das principais diferenças entre o Robonaut e o NAO é a ausência de pernas, então o Robonaut não é capaz de se mover caminhando. Por outro lado, o Robonaut pode ser acoplado a um Segway RMP, um veículo de transporte motorizado semelhante à uma cadeira de rodas. O Segway RMP é controlado por um conjunto de pedais.

2 FERRAMENTAS MATEMÁTICAS E DE ENGENHARIA

As ferramentas utilizadas neste trabalho e descritas neste capítulo definem os detalhes no funcionamento dos vários processos necessários para a execução do sistema de teleoperação desenvolvido neste trabalho.

2.1 REPRESENTAÇÃO NO ESPAÇO TRIDIMENSIONAL

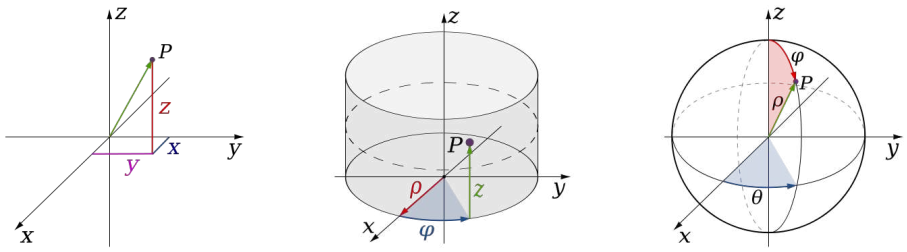
Nesta seção são feitas definições matemáticas de sistemas de coordenadas, matrizes de rotação e de transformação homogênea, quatérnios e quatérnios duais. Essas ferramentas são importantes para representar as partes do corpo do teleoperador no espaço ou como fazer a transformação de uma posição até a outra até o outra. As definições matemáticas desta seção foram definidas em [5] e [6].

2.1.1 Sistemas de coordenadas, posição e orientação

As várias partes do corpo humano e de robôs humanoides podem ser representadas por objetos no espaço tridimensional, geralmente referido como $\mathbb{R}^3$, e estas partes estão ligadas umas às outras. Para atingir o propósito deste trabalho, é possível representar, de forma simplificada, todo o corpo humano por pontos chave específicos e adotar uma posição relativa de um ponto até o outro.

O espaço tridimensional $\mathbb{R}^3$, assim como qualquer outro $\mathbb{R}^n$, é um espaço vetorial [5], em que são definidas as suas propriedades e possíveis operações sobre os vetores desses espaços. Exemplos de propriedades seriam comutatividade, associatividade, existência de zero, entre outras. Exemplos de operações seriam adição, produto escalar, produto vetorial, entre outras.

No espaço tridimensional, o elemento mais simples que pode ser representado é um ponto. Um ponto pode ser representado por três sistemas de coordenadas, cada um com três parâmetros. No sistema de coordenadas cartesianas ou retangulares (Fig. 2.1(a)), um ponto pode ser representado pelos valores x, y e z em cada um dos três eixos ortogonais, cada um deles representando uma das dimensões do espaço. No sistema de coordenadas cilíndricas (Fig. 2.1(b)), um ponto pode ser representado por uma distância radial à origem ρ, um ângulo polar φ entre o ponto e o eixo X, e a altura z. No sistema de coordenadas esféricas (Fig. 2.1(c)), um ponto pode ser representado por uma distância radial à origem r, um ângulo polar ou de colatitude φ entre o eixo Z e o segmento de reta OP (O é a origem e P é o ponto a ser representado), e um ângulo de azimute ou de longitude θ entre o eixo X e a reta projetada de OP no plano XY.



(a) Coordenadas cartesianas ou retangulares (b) Coordenadas cilíndricas (c) Coordenadas esféricas

Figure 2.1: Sistemas de coordenadas

Além de pontos, outros objetos também podem ser representados no espaço como retas, planos, circunferências, círculos, esferas, polígonos, poliedros, etc. Estes objetos, compostos por infinitos pontos, possuem algum tipo de extensão em pelo menos uma dimensão e orientação, que um ponto não possui. A orientação pode ser representada pelos ângulos de Euler row (φ), pitch (θ) e yaw (ψ). Com a possibilidade de representar a orientação, assim como existe o conceito de posição relativa, também é possível utilizar uma orientação relativa entre dois referenciais. Dessa forma, um objeto que é representado com determinados valores por algum sistema de coordenadas e com determinada orientação, apresentará outros valores de coordenadas e de orientação quando for observado a partir de outro referencial. Para transformar a orientação de um referencial a outro, é de se pensar que é suficiente fazer apenas as rotações nos ângulos. Porém, quando ocorrem transformações que usam mais de um dos três ângulos, ao fazê-las em um ângulo de cada vez deve-se atentar à ordem correta em que essas transformações ocorrem, já que estas operações não são comutativas (Fig. 2.2).

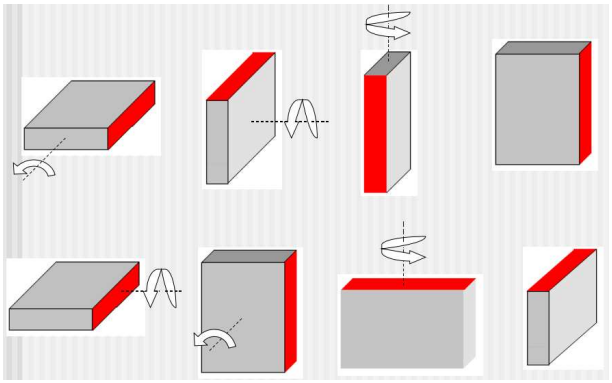


Figure 2.2: Rotação não comutativa

2.1.2 Matrizes de Rotação e Matrizes Transformação Homogênea

Uma alternativa que se usa para realizar as mudanças de orientação são as matrizes de rotação. Uma matriz de rotação rotaciona um vetor representado em coordenadas cartesianas. A transformação mais simples é uma rotação básica em torno de um dos eixos de coordenadas cartesianas. As matrizes

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sen\theta \\ 0 & sen\theta & cos\theta \end{bmatrix} \tag{2.1}$$

$$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix} \tag{2.2}$$

$$R_z(\theta) = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

representam as rotações em cada um dos eixos ortogonais em coordenadas cartesianas, sendo θ positivo no sentido anti-horário. Considerando um ângulo de rotação para cada eixo, a rotação resultante em três eixos pode ser descrita por

$$R(\alpha, \beta, \gamma) = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{2.4}$$

lembrando que a ordem do produto importa, já que essas transformações não são comutativas, assim como são as multiplicações de matrizes. Além disso, a ordem das rotações não necessariamente deve ser em torno de x, de y e de z, e sim na ordem em que for necessária para representar a transformação desejada, ou até mesmo em um outro eixo qualquer que pode ser decomposto nas três dimensões. Também é possível representar transformações que usam rotação e translação usando matrizes 4x3, chamadas de Matrizes de Transformação Homogênea. Uma Matriz de Transformação Homogênea, envolvendo rotação e translação, é dada por:

$$T = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & d_x \\ R_{yx} & R_{yy} & R_{yz} & d_y \\ R_{zx} & R_{zy} & R_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Com essa matriz, é possível transformar um ponto $\vec{p} = (p_x, p_y, p_z)^T$ para um outro ponto $\vec{p'} = (p'_x, p'_y, p'_z)^T$através da operação:

$$\vec{p'} = T\vec{p} \Rightarrow \begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & d_x \\ R_{yx} & R_{yy} & R_{yz} & d_y \\ R_{zx} & R_{zy} & R_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{2.6}$$

O uso de matrizes de rotação ou de transformação (compondo rotação e translação) parece ser uma ferramenta boa o suficiente para lidar com rotações, porém esta ferramenta matemática apresenta alguns problemas. As vantagens do uso de matrizes incluem elas serem ensinadas cedo no curso superior ou até mesmo no ensino médio, o que a torna uma ferramenta mais familiar, além de muitos algoritmos serem implementados com o uso de matrizes. Em contrapartida, matrizes de transformação requerem pelo menos 12 parâmetros para representar estruturas com apenas 6 graus de liberdade (ou

DOF, do inglês degrees of freedom); a parte rotacional (3x3) da matriz de transformação é composta de colunas ortogonais, cujos valores podem sofrer grandes variações e trazer desvios e proporções indesejadas, sendo possível ter que renormalizar a matriz usando o método de Gram-Schmidt, que é computacionalmente custoso; fazer a interpolação entre matrizes é difícil; não é fácil visualizar uma matriz e as componentes angulares de cada eixo sobre os quais ela irá rotacionar e transladar; entre outros problemas.

2.1.3 Quatérnios e Quatérnios Duais

Outra alternativa para representar rotações no espaço tridimensional é o uso de quatérnios[6]. Quatérnios foram introduzidos por Hamilton em 1866 como uma expansão dos números complexos. Um quatérnio é definido como

$$\mathbf{q} = w + (xi + yj + zk) \tag{2.7}$$

sendo w, x, y e z números reais, enquanto i, j e k são componentes imaginárias puras unitárias. As propriedades das componentes imaginárias são:

$$i^2 = j^2 = k^2 = -1$$

$$ij = k, \qquad ji = -k$$

$$jk = i, \qquad kj = -i$$

$$ki = j, \qquad ik = -j$$

Também é possível representar quatérnios em duas componentes, uma vetorial ($\vec{v} = xi + yj + zk$) e uma escalar (w).

$$\mathbf{q} = (w, \vec{v}) \tag{2.8}$$

As operações aritméticas com quatérnios são:

• Multiplicação por escalar: $s\mathbf{q} = (sw, s\vec{v})$, sendo s um valor escalar

• Adição: $\mathbf{q}_1 + \mathbf{q}_2 = (w_1 + w_2, \vec{v_1} + \vec{v_2})$

• Multiplicação (produto interno ou escalar): $\mathbf{q}_1\mathbf{q}_2 = \mathbf{q}_2\mathbf{q}_1 = w_1w_2 + x_1x_2 + y_1y_2 + z_1z_2$, ou seja, o resultado do produto escalar é um número real

• Multiplicação (produto externo ou vetorial): $\mathbf{q}_1 \times \mathbf{q}_2 = (w_1w_2 - \vec{v_1}\vec{v_2}, w_1\vec{v_2} + w_2\vec{v_1} + (\vec{v_1} \times \vec{v_2})) \neq \mathbf{q}_2 \times \mathbf{q}_1$

• Conjugado: $\mathbf{q}^* = (w, -\vec{v})$

• Módulo: $\|\mathbf{q}\| = \mathbf{q}\mathbf{q}^*$

Um quatérnio unitário tem sua norma $\|\mathbf{q}\| = 1$. O quatérnio unitário é usado para representar uma rotação de um ângulo θ em torno de um eixo unitário $\mathbf{n}$ no espaço tridimensional.

$$\mathbf{q} = \left(cos\left(\frac{\theta}{2}\right), \mathbf{n} sen\left(\frac{\theta}{2}\right)\right) \tag{2.9}$$

É possível combinar quatérnios com números duais. A unidade dual $\boldsymbol{\epsilon}$ possui as seguintes propriedades:

$$\boldsymbol{\epsilon} \neq 0 \qquad \boldsymbol{\epsilon}^2 = 0$$

Um número dual possui uma parte dual, que é multiplicada pela unidade dual, somada a uma parte real. Da mesma forma, um quatérnio dual é composto por oito elementos ou dois quatérnios, um deles real e o outro dual

$$\underline{\mathbf{q}} = \mathbf{q}_r + \mathbf{q}_d\boldsymbol{\epsilon} \tag{2.10}$$

sendo que $\mathbf{q}_r$ e $\mathbf{q}_d$ são quatérnios. A razão de se combinar quatérnios com números duais é que, enquanto quatérnios podem representar apenas rotações, quatérnios duais podem representar tanto rotações como translações. Combinando as operações algébricas associadas aos quatérnios com a unidade dual, é possível descrever as operações aritméticas de quatérnios duais.

• Multiplicação por escalar: $s\underline{\mathbf{q}} = s\mathbf{q}_r + s\mathbf{q}_d\boldsymbol{\epsilon}$

• Adição: $\underline{\mathbf{q}}_1 + \underline{\mathbf{q}}_2 = \mathbf{q}_{r1} + \mathbf{q}_{r2} + (\mathbf{q}_{d1} + \mathbf{q}_{d2})\boldsymbol{\epsilon}$

• Conjugado: $\underline{\mathbf{q}}^* = \mathbf{q}_r^* + \mathbf{q}_d^*\boldsymbol{\epsilon}$

• Módulo: $\left\|\underline{\mathbf{q}}\right\| = \underline{\mathbf{q}\mathbf{q}}^*$

• Condição unitária: $\left\|\underline{\mathbf{q}}\right\| = 1 \quad \text{e} \quad \mathbf{q}_r^*\mathbf{q}_d + \mathbf{q}_d^*\mathbf{q}_r = 0$

Um quatérnio dual unitário é capaz de representar qualquer transformação rígida de rotação e translação. A informação da transformação rígida de rotação e translação no quatérnio dual unitário é:

$$\mathbf{q}_r = \mathbf{r} \qquad \mathbf{q}_d = \frac{1}{2}\mathbf{t}\mathbf{r}$$

sendo que $\mathbf{r}$ é o quatérnio unitário representando a rotação e $\mathbf{t}$ é o quatérnio descrevendo a translação representado pelo vetor $\mathbf{t} = (0, t_x, t_y, t_z)$.

Quatérnios duais podem representar rotação pura da mesma forma que quatérnios definindo a parte dual como zero.

$$\underline{\mathbf{q}}_r = \left[cos\left(\frac{\theta}{2}\right), \mathbf{n}_x sen\left(\frac{\theta}{2}\right), \mathbf{n}_y sen\left(\frac{\theta}{2}\right), \mathbf{n}_z sen\left(\frac{\theta}{2}\right)\right] [0,0,0,0] \tag{2.11}$$

Para representar uma translação pura sem rotação, a parte real pode ser definida como uma identidade e a parte dual representando a translação.

$$\underline{\mathbf{q}}_t = [1,0,0,0]\left[0, \frac{t_x}{2}, \frac{t_y}{2}, \frac{t_z}{2}\right] \tag{2.12}$$

Combinando as transformações de rotação e de translação em um único quatérnio dual unitário para representar a rotação seguida da translação, obtém-se:

$$\underline{\mathbf{q}} = \underline{\mathbf{q}}_t \underline{\mathbf{q}}_r \tag{2.13}$$

A seguinte operação aritmética define como transformar o ponto $\vec{p}$, representado no espaço, usando um quatérnio dual unitário:

$$\mathbf{p}' = \underline{\mathbf{q}}\vec{p}\underline{\mathbf{q}}^* \tag{2.14}$$

2.2 EQUIPAMENTOS PARA SENSOREAMENTO E ATUAÇÃO DO SISTEMA

2.2.1 Sensores inerciais e determinação da posição e da orientação

Com os sistemas de coordenadas, ângulos de Euler, matrizes de transformação, quatérnios e quatérnios duais é possível representar objetos no espaço tridimensional, assim como a movimentação destes objetos no espaço. Porém, ainda é necessário obter as informações no mundo real sobre a posição das partes do corpo humano que devem ser representadas. Uma possível forma de se obter dados sobre a posição e orientação de um objeto real é através de sensores inerciais, que geralmente são encontrados em dispositivos que possuem uma unidade de medição inercial (ou IMU, do inglês inertial measurement unit).

IMUs[7] são dispositivos que mensuram a aceleração, a velocidade angular e, muitas vezes, o campo magnético cercante de um corpo, através de acelerômetros, giroscópios e magnetômetros. Com o uso dessas medidas é possível determinar por meio de cálculos o deslocamento linear e angular de um corpo, assim como a sua orientação em relação ao planeta. Se a posição inicial do corpo é conhecida, também é possível determinar a posição e orientação instantâneas do corpo enquanto ele se move no mundo. IMUs são encontradas em muitos veículos aéreos, terrestres e aquáticos, assim como em drones, robôs, smartphones, dispositivos de GPS, videogames, entre outros.

Acelerômetro é um dispositivo capaz de medir a aceleração própria, que é diferente do conceito de aceleração (taxa de variação da velocidade no tempo). O acelerômetro de um objeto parado na Terra idealmente irá medir a aceleração da gravidade (9,81 m/s² para baixo), mas quando o objeto sofre queda livre, este acelerômetro idealmente mediria zero de aceleração em qualquer direção. Um exemplo que demonstra o funcionamento do acelerômetro é o de um objeto pendurado no teto de um veículo. Quando o veículo está parado, o objeto fica para baixo, já que a única força atuante é o peso do objeto. Quando o veículo se move para frente, o objeto se move para trás em relação ao veículo, devido à inércia do objeto, e o ângulo da linha que segura o objeto representa uma proporção entre o peso do objeto e a aceleração que o objeto está sofrendo devido ao movimento do veículo. Quando o veículo freia, o objeto se move para frente em relação ao veículo, assim como o objeto se move para os lados em relação ao veículo quando o veículo realiza uma curva. Atualmente são produzidos modelos de acelerômetros muito mais sofisticados usando, por exemplo, sistemas microeletromecânicos.

Girômetro[8] é um dispositivo que mede a velocidade angular. Existem muitas aplicações para girômetros e hoje podem ser encontrados em vários dispositivos como câmeras, drones, smartphones e em aplicações militares e aeroespaciais. Aplicações comuns de girômetros são: navegação, controle de trajetória de um objeto em movimento, estabilização do sistema em torno de um eixo ou em relação ao solo, entre outros. Existem muitas variações de girômetros que utilizam diversas tecnologias como sistemas microeletromecânicos (MEMS), fibras óticas, lasers, entre outros.

Os sensores inerciais presentes nas IMUs de dispositivos eletrônicos portáteis são geralmente do tipo microeletromecânicos encontrados dentro de chips muito pequenos. Com a utilização de algum dispositivo eletrônico portador de uma IMU acoplado em alguma parte do corpo de uma pessoa, é possível capturar as informações de aceleração e velocidade angular daquela parte do corpo. Com essas informações e sabendo a posição inicial da parte do corpo é possível determinar, por integração, a posição e a orientação instantâneas no espaço tridimensional.

Assim como qualquer outro sensor, os sensores inerciais presentes nas IMUs possuem erros em suas medidas[9]. Alguns dos tipos de erros encontrados nestes sensores são:

• Viés constante: um erro constante com a taxa, resultando em um erro que aumenta linearmente com o tempo com a integração para a obtenção da medida (posição ou ângulo);

• Ruído branco termomecânico: erro variável aleatório, maior que a taxa de amostragem do sensor;

• Ruído de oscilação: normalmente são observados em baixas frequências em dispositivos eletrônicos.

Se os sensores fossem perfeitos (fossem livre de erros, tivessem taxa de amostragem e resolução infinitas), teoricamente não haveria problemas em obter a posição e a orientação dos objetos do mundo real. Uma possível solução para atenuar os erros de medida dos sensores é com a aplicação de filtros. Exemplos de filtros que podem ser aplicados para tratar os dados dos sensores são: filtro de média móvel[10] e filtro de Kalman[11].

2.2.2 Robô humanoide NAO

Desenvolvido pela empresa francesa Aldebaran Robotics, o NAO[12] (Fig. 2.3) é um robô humanoide autônomo e programável, com sua primeira versão lançada em 2006. A versão do NAO utilizada neste trabalho é a V4 H25 (4ª versão com 25 graus de liberdade)[1]. Além das juntas para mover as partes dos membros e da cabeça, este robô possui sensores em cada uma das juntas para indicar o ângulo da junta, uma bateria íons de lítio de 2.25 Ah, 21.6V e 48.6 Wh que permite uma autonomia teórica de até em torno de 60 minutos com uso ativo, uma IMU com girômetro e acelerômetro, 4 sensores ultrassônicos no peito, sensores de força resistivos nas solas dos pés, um bumper (tipo de botão) em cada pé, um botão no peito com o logo da Aldebaran e sensores capacitivos nas mãos. Em sua cabeça também existem 3 sensores capacitivos, sensores infravermelhos nos olhos, duas câmeras, 4 microfones, duas caixas de som e uma placa-mãe com processador Intel Atom Z530

[1]NAO H25 — Aldebaran 2.1.4.13 documentation: mais informações podem ser encontradas em http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html. Acesso em: 08 dez. 2016

1.6 GHz, 1 GB de RAM, 2 GB de memória flash (onde é instalado o seu sistema operacional) e 8 GB de memória Micro SDHC. O robô também possui LEDs nos olhos, nos sensores capacitivos da cabeça, no botão do peito e nos pés. Ele pode se conectar por cabo Ethernet ou WiFi e possui uma porta USB. O sistema operacional utilizado pelo NAO é uma distribuição de Linux baseada em Gentoo chamada de NAOqi OS. A Aldebaran fornece a documentação e as bibliotecas do NAOqi, que possuem funções para que permitem acessar os sensores e os atuadores do NAO. É possível fazer programas, em linguagens C++ e Python, que o controlam remotamente a partir de um computador ou que são executados dentro do robô.

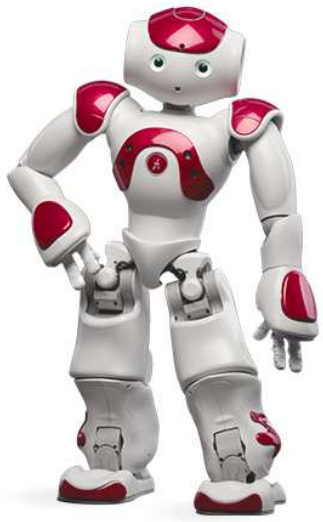


Figure 2.3: Robô humanoide NAO, desenvolvido pela Aldebaran Robotics (Fonte: Aldebaran)

2.2.3 Kinect

Kinect[13] é uma linha de dispositivos de captura de movimento criada pela Microsoft para Xbox 360, Xbox One e computadores. O primeiro Kinect (Fig. 2.4), desenvolvido inicialmente para o Xbox 360, possui um projetor e um sensor de profundidade infravermelho que são usados para gerar um mapa tridimensional da região capturada. Esse mapa tridimensional pode ser visualizado no formato de nuvem de pontos (Fig. 2.5). O Kinect é conectado ao computador por uma porta USB.

Figure 2.4: Kinect para Xbox 360

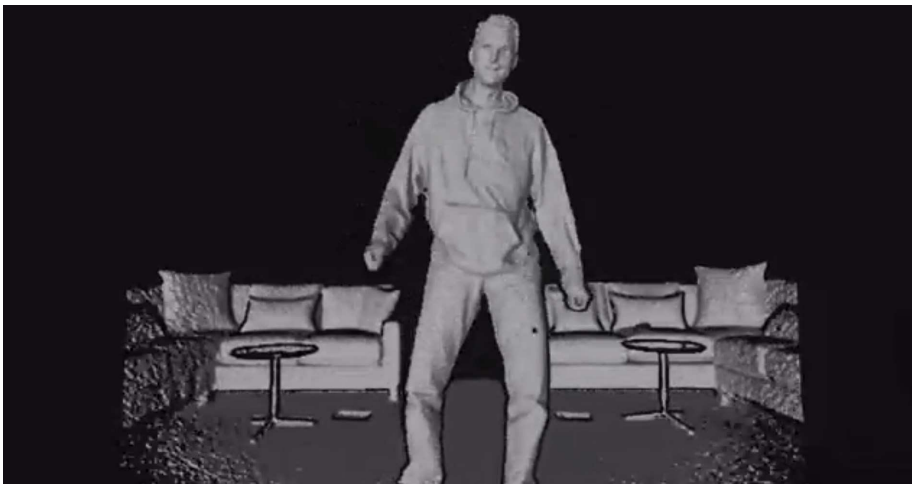


Figure 2.5: Nuvem de pontos gerada pela captura do Kinect

Dentro da nuvem de pontos gerada pela captura do Kinect, é possível segmentar um corpo humano com as suas partes e gerar um esqueleto virtual (Fig. 2.6) que inclui cabeça, tronco e membros.

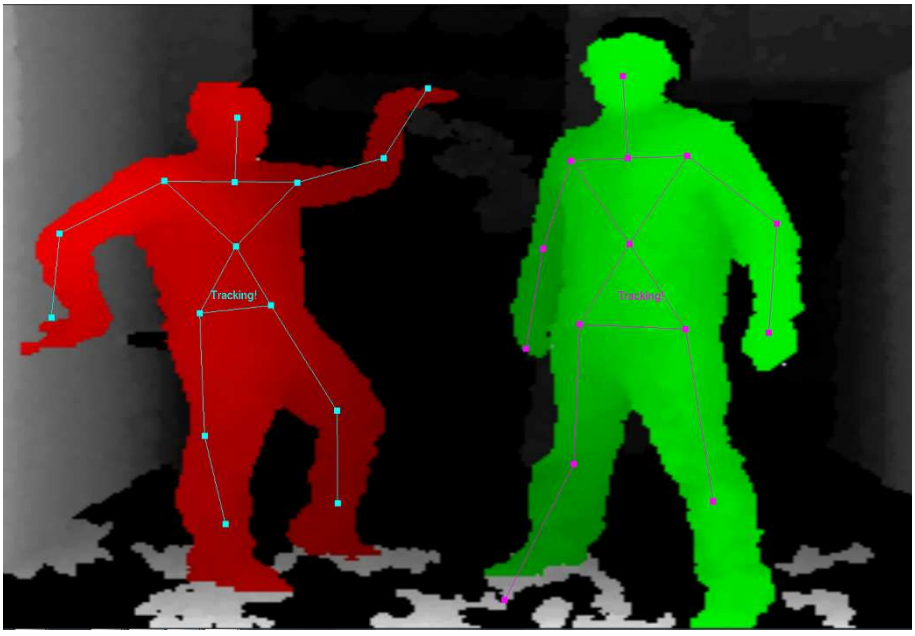


Figure 2.6: Segmentação do corpo humano na nuvem de pontos com esqueleto virtual (Fonte: The Serious Computer Vision Blog)

A razão de se utilizar o Kinect é de capturar os movimentos do corpo além de utilizar IMUs. O problema de se utilizar IMUs para capturar os movimentos do corpo é que, idealmente, cada parte do corpo que se deseja capturar os movimentos precisa ter uma IMU acoplada a ela. Com o Kinect, é possível capturar os movimentos de todo o corpo usando apenas um dispositivo. Dentro do mapa tridimensional gerado pelos dados de captura do Kinect, é possível rastrear um esqueleto humano virtual. Este esqueleto virtual é composto por várias juntas e conexões entre as juntas, que representam as diversas partes de um corpo humano.

Um dos problemas de se usar o Kinect é que ele nem sempre apresenta uma boa precisão, prin-

cipalmente com relação à orientação. Então para determinar a orientação de partes específicas do corpo, como a cabeça, ainda é necessário fazer o uso de IMUs acopladas à essas partes.

2.2.4 Myo

O bracelete Myo[14] (ou em inglês, Myo armband), fabricado pela Thalmic Labs, é um dispositivo que permite controlar outros dispositivos através de gestos e movimentos do braço em que ele é usado. O Myo (Fig. 2.7), além de possuir uma IMU embutida, possui 8 sensores eletromiográficos (EMG) que captam a atividade elétrica nos músculos do braço (Fig. 2.8). Dessa forma, o Myo é capaz de reconhecer gestos feitos com a mão. Sabendo que tipo de gesto o usuário está fazendo com uma de suas mãos, é possível transmitir esse gesto para que o robô possa representá-lo. Neste trabalho o Myo é utilizado para capturar o gesto de abrir e fechar a mão, dessa forma podendo controlar o abrir e fechar dos dedos do robô. Um exemplo de aplicação do Myo para algo semelhante, porém muito mais complexo do que apenas abrir e fechar os dedos de uma mão, é visto no trabalho de Kuiken et al.[15], em que o Myo é utilizado por uma pessoa com algum membro superior amputado para controlar um braço prostético.

As informações capturadas pelo Myo são transmitidas a um dongle USB conectado ao computador por uma comunicação Bluetooth dedicada entre o dongle e o Myo.

Figure 2.7: Myo armband (Fonte: Thalmic Labs)

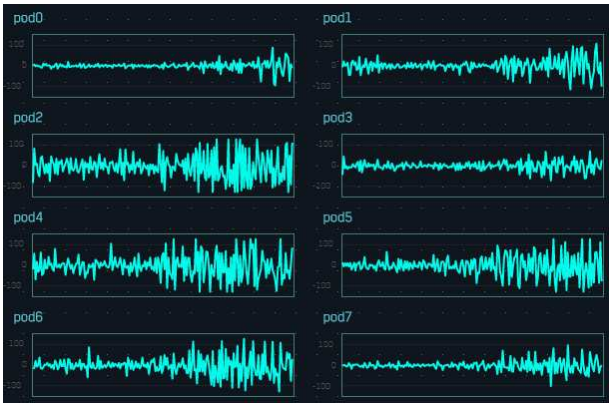


Figure 2.8: Visualização de dados dos sensores EMG do Myo

Figure 2.9: Raspberry Pi 2

2.2.5 Raspberry Pi

Raspberry Pi é uma série de computadores de placa única desenvolvidos pela Raspberry Pi Foundation. Os modelos de Raspberry Pi[2] possuem algum processador ARM, porta HDMI para saída de vídeo, portas USB, porta de saída de áudio analógica, armazenamento por cartão de memória SD ou MicroSD, portas GPIO e a maioria dos modelos possui porta Ethernet. Sendo capaz de rodar sistemas operacionais como o Raspbian (uma variação de Debian para a Raspberry Pi), é possível usar estes computadores como um computador pessoal, mas devido ao tamanho e o baixo consumo de energia em relação a um computador normal (é alimentado por microUSB 5V e consome entre 160mA e 800mA dependendo do modelo), essas placas também podem ser utilizadas como sistemas embarcados.

Neste trabalho, é utilizada a Raspberry Pi 2 (Fig. 2.11) para obter as imagens de uma webcam USB e fazer a transmissão delas por rede para o computador. Um exemplo de aplicação de sistema embarcado para capturar imagens é feito por Senthilkumar at el. em [16], em que as imagens capturadas são processadas por um programa de reconhecimento que é executado pela Raspberry Pi.

2.2.6 Realidade virtual e telepresença

Para que o sistema de teleoperação funcione como desejado, não é suficiente apenas capturar e transmitir os movimentos do usuário para o robô. Também é necessário mostrar para o usuário as informações do ambiente em que o robô teleoperado se encontra para que o teleoperador se sinta imerso neste ambiente, como se o corpo do robô fosse o seu próprio corpo. Essa tarefa é realizada utilizando as tecnologias relacionadas à realidade virtual. Realidade virtual[17] é uma tecnologia computacional que recria um ambiente (imaginário ou real) e simula a presença física do usuário permitindo a interação do usuário com este ambiente.

Semelhante à realidade virtual, também existe o conceito de telepresença, que permite à pessoa sentir que está presente e dar a aparência de que está presente em um ambiente diferente da sua localização real. A principal diferença entre realidade virtual e telepresença é o tipo de ambiente presenciado pela pessoa e com a qual ela interage. Na realidade virtual, o ambiente é virtual ou simulado e na telepresença o ambiente é real. Um exemplo muito comum do uso de telepresença é a

[2]Raspberry Pi Products - Where to Buy Raspberry Pi: mais informações podem ser encontradas em https://www.raspberrypi.org/products/. Acesso em 08 dez. 2016

videoconferência.

O sistema de teleoperação controla um robô humanoide real no mundo real, ou seja, é um sistema de telepresença, então a realidade presenciada pelo usuário do sistema será um ambiente real, porém do ponto de vista do robô. A captura das informações do ambiente em que se encontra o robô é feito por uma câmera de vídeo acoplada ao robô (como se fossem os "olhos" do robô), e a imagem desta câmera é transmitida em tempo real para o usuário do sistema de teleoperação e apresentada ao usuário em um formato de realidade virtual em um dispositivo do tipo HMD (do inglês Head-mounted Display). O HMD (Fig. 2.10) escolhido para a execução deste trabalho é um smartphone com sistema operacional Android inserido em um Cardboard (Fig. 2.11). O Cardboard[18], criado pela Google, é um equipamento de baixo custo feito de papelão que transforma um smartphone em um HMD, com o objetivo de fornecer uma experiência de realidade virtual com baixo custo. A imagem a ser apresentada em um formato de realidade virtual significa que a imagem de uma câmera será mostrada no HMD para os dois olhos, e cada olho recebe uma imagem individual.

Devido à estrutura dos HMD como o Cardboard e à forma em que ele é utilizado, a proximidade da tela com os olhos dão ao usuário a impressão de que a pessoa enxerga um ambiente por um outro ponto de vista, normalmente como se estivesse presenciando o ambiente no lugar de outra pessoa, objeto ou personagem. A sensação de imersão no ambiente se torna maior quando os movimentos do usuário correspondem ao que o usuário enxerga na tela (por exemplo, quando o usuário move a cabeça para o lado, se abaixa ou move os seus braços enxergando os braços virtuais se movendo de acordo).

(a) HMD sendo utilizado por uma pessoa (Fonte: Dynamic Graphics Project lab of the University of Toronto)

(b) Visão através das lentes de um HMD (Fonte: Paracosm)

Figure 2.10: Head-mounted display

(a)

(b)

Figure 2.11: Google Cardboard

2.3 FERRAMENTAS DE SOFTWARE

2.3.1 Robot Operating System

Robot Operating System[19] (ROS) é um meta sistema operacional de código aberto para robôs. Ele providencia os serviços que são esperados de um sistema operacional , incluindo abstração de hardware, controle de dispositivos em baixo nível, passagem de mensagem entre processos e gerenciamento de pacotes. Ele também possui ferramentas e bibliotecas para obter, construir, escrever e executar códigos através de múltiplos computadores. O núcleo do sistema ROS, junto com ferramentas e bibliotecas úteis são lançados regularmente como uma distribuição de ROS, semelhante à uma distribuição de Linux, que providencia um conjunto de softwares compatíveis para que outras pessoas possam utilizar e construir sobre eles. A distribuição de ROS utilizada neste trabalho é a Indigo. O ROS possui três níveis de conceito: o nível Filesystem (arquivos de sistema), o nível Computation Graph (grafo de computação) e o nível Community (comunidade).

O nível Filesystem inclui os arquivos que ficam armazenados em disco como pacotes, meta pacotes, manifestos de pacotes, repositórios, mensagens e serviços.

• Pacotes: São a principal unidade para organizar softwares no ROS. Pode conter processos ROS em tempo de execução (também chamados de nós ou nodes), uma biblioteca dependente de ROS, conjuntos de dados, arquivos de configuração, ou qualquer outra coisa que possa ser agrupada de forma organizada e seja útil. Pacotes são o item de construção e item de lançamento mais atômico no ROS;

• Meta pacotes: Servem para representar um grupo de pacotes relacionados;

• Manifestos de pacotes: Providenciam os metadados sobre um pacotes;

• Repositórios: Coleção de de pacotes que compartilham um sistema VCS (version control system) em comum. Pacotes que compartilham um VCS compartilham a mesma versão e podem ser lançados juntos;

• Mensagens - Arquivo com descrições de mensagens que definem as estruturas de dados nas mensagens enviadas no ROS;

• Serviços - Arquivo com descrições de serviços definem as estruturas de dados das requisições e respostas usadas pelos serviços no ROS.

O nível Computation Graph é a rede ponto a ponto dos processos do ROS que estão processando dados juntos. Os conceitos básicos de Computation Graph são nodes (ou nós), Master, Parameter Server, mensagens, serviços, tópicos e bags, sendo que todos eles fornecem dados de formas diferentes.

• Nós: São processos que efetuam computacão. O ROS é projetado para ser modular; um sistema de controle de um robô geralmente utiliza muitos nós. Por exemplo, um nó controla um laser medidor de distância, outro nó controla os motores das rodas, outro nó faz a localização, outro nó faz o planejamento de trajetória, outro nós faz a visão gráfica do sistema, e assim por diante. Um nó do ROS é escrito com o uso de uma ROS client library (que contém muitos dos conceitos do ROS na forma de código) como roscpp (para C++) ou rospy (para Python);

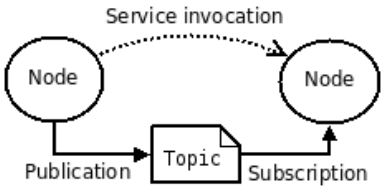


Figure 2.12: Invocação de serviço e transmissão de mensagem no ROS (Fonte: ROS Wiki)

• Master: Providencia o registro do nome e procura pelo resto do grafo. Sem o Master, os nós não seriam capazes de encontrar uns aos outros, trocar mensagens ou invocar serviços;

• Parameter Server: Permite que os dados sejam armazenados por chaves em um local central. Atualmente é uma parte do Master;

• Mensagens: Nós se comunicam uns com os outros passando mensagens. A mensagem é simplesmente uma estrutura de dados, contendo campos tipados. Tipos primitivos padrões (inteiro, ponto flutuante, booleano, etc.) são suportados, assim como são arrays de tipos primitivos. Mensagens pode incluir arbitrariamente estruturas aninhadas e arrays (assim como structs em C);

• Tópicos: Mensagens são encaminhadas por um sistema de transporte com semântica de publish (publicar) / subscribe (inscrever). Um nó envia uma mensagem publicando ela para um dado tópico. O tópico é um nome que é usado para identificar o conteúdo de uma mensagem. Um nó que está interessado em um certo tipo de dado irá se inscrever no tópico apropriado. Podem existir múltiplos publishers e subscribers concorrentes para um único tópico e um único nó pode publicar ou se inscrever em muitos tópicos. Em geral, publishers e subscribers não estão cientes da existência um do outro. A ideia é desacoplar a produção de informação do seu consumo;

• Serviços: O modelo de publish / subscribe é um paradigma de comunicação muito flexível, mas sua forma de transporte de via única de muitos para muitos não é apropriada para interações de requisições e respostas, que são geralmente requeridas em um sistema distribuído. Requisições e respostas são feitas através de serviços, que são definidos por um par de estruturas de mensagens: uma para a requisição e outra para a resposta. O nó que providencia oferece um serviço sob um nome e um cliente usa o serviço enviando a mensagem requisitada e esperando pela resposta;

• Bags: Bags são um formato para salvar e reproduzir os dados das mensagens do ROS. Bags são um mecanismo importante para armazenar dados, tais como dados de sensores, que podem ser difíceis de coletar, mas são necessários para desenvolver e testar algoritmos.

Os conceitos do nível Community são recursos do ROS que permitem que comunidades separadas troquem softwares e conhecimento entre elas. Esses recursos incluem distribuições, repositórios, a ROS Wiki, Blog, entre outros.

2.3.2 Pacotes do ROS skeleton_markers e openni_tracker

O pacote skeleton_markers[3] utiliza os drivers do Kinect para Linux para acessar os dados vindos do Kinect. O skeleton_markers é mostrar as juntas do esqueleto virtual retornadas pelo pacote openni_tracker. O pacote utiliza dois nós para ler o esqueleto rastreado de duas formas diferentes: um dos nós se inscreve no tópico Skeleton retornado pelo nó skeleton_tracker incluído neste pacote e um outro nó pode ler as transformadas diretamente do pacote openni_tracker.

O pacote openni_tracker segmenta o corpo humano na nuvem de pontos do Kinect e transmite os frames do esqueleto virtual.

2.3.3 Pacote do ROS ros_myo

Este pacote do ROS é responsável por lidar com os dados dos sensores EMG do Myo. O ros_myo[4] cria um nó que publica os dados puros do Myo tanto na forma de mensagens do ROS. É possível se inscrever no tópico para acessar a essas mensagens e utilizá-las nas arquiteturas padrões do ROS.

2.3.4 NAOqi

NAOqi [20] é o nome do principal software que é executado no NAO e o controla. O framework NAOqi é o framework de programação usado para programar os robôs da Aldebaran. Ele é capaz de lidar com várias aplicações necessitadas pela robótica como paralelismo, sincronização, entre outros. Este framework pode ser utilizado em outras plataformas além do próprio sistema operacional do robô, em Windows, Linux e MacOS, nas linguagens C++ e Python. O NAOqi (Fig. 2.13) é um software mediador (broker) que decide quais das várias bibliotecas ele deve carregar no robô a partir de um arquivo de preferências chamado autoload.ini. Cada biblioteca possui um ou mais módulos, que permitem utilizar as várias funcionalidades do robô (ALMotion para mover as juntas, ALLeds para controlar os LEDs, entre outros). Tipicamente, um módulo é uma classe dentro de uma biblioteca.

Cada módulo (Fig. 2.14) possui os seus próprios métodos que permitem executar as diversas ações específicas no robô, como comandar um movimento de caminhada ou obter os dados de algum sensor. Os módulos podem ser locais ou remotos. Módulos locais são dois ou mais módulos carregados no mesmo processo, então eles são capazes de se comunicar usando apenas um broker e compartilhar variáveis e métodos sem utilizar rede ou serialização. Módulos remotos são aqueles que se comunicam por rede, isso significa que esses módulos precisam de um broker para se comunicarem com outros módulos. O broker é responsável pela comunicação em rede, permitindo que os métodos dos módulos atachados sejam chamados fora do processo.

Um proxy é um objeto que se comporta como o módulo que ele representa. Através do proxy, é utilizado o IP e a porta de um broker, que permite o acesso ao módulo equivalente ao proxy no broker

[3] GitHub - pirobot/skeleton_markers: mais informações podem ser encontradas em https://github.com/pirobot/skeleton_markers. Acesso em: 08 dez. 2016

[4] GitHub - roboTJ101/ros_myo: mais informações podem ser encontradas em https://github.com/roboTJ101/ros_myo. Acesso em: 08 dez. 2016

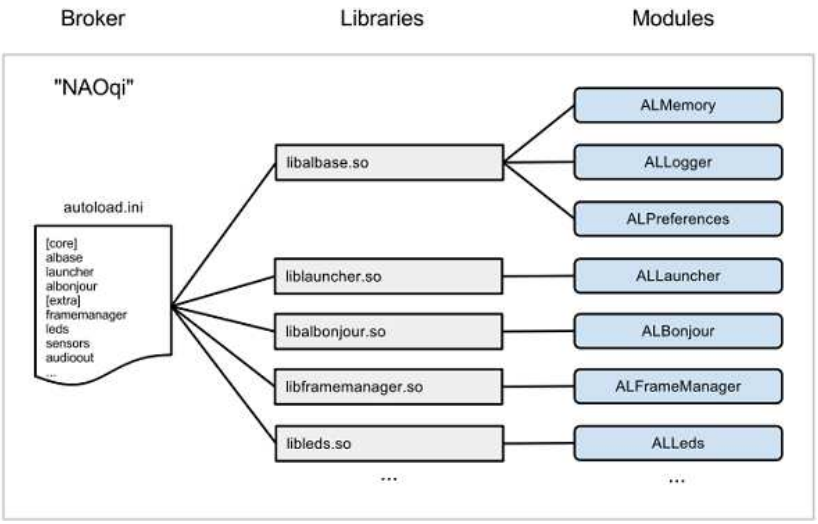


Figure 2.13: NAOqi acessa as bibliotecas no autoload.ini que contém os módulos (Fonte: Aldebaran)

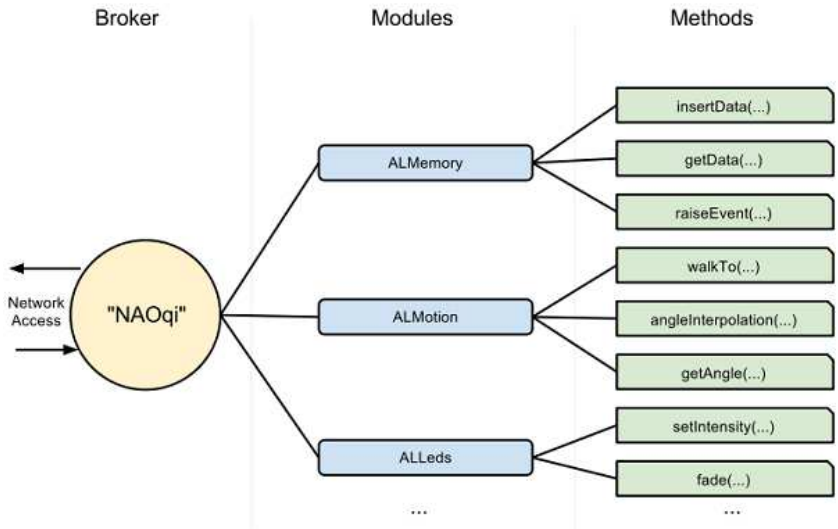


Figure 2.14: Métodos atachados aos módulos atachados ao broker que faz a comunicação em rede (Fonte: Aldebaran)

em que se deseja conectar. Uma conexão entre um proxy e um broker abre uma comunicação de via única, ou seja, o proxy pode acessar todos os módulos do broker, mas os módulos do broker não podem acessar ao módulo que o proxy pertence. Neste trabalho são utilizados métodos de módulos e proxys para gerar os comandos que o robô irá executar e capturar os dados dos sensores do robô para a obtenção dos resultados.

2.3.5 motionEyeOS

O motionEyeOS[21] é um sistema operacional desenvolvido para computadores de placa única como a Raspberry Pi. Ele é uma distribuição de Linux customizada com o principal propósito de capturar as imagens de uma câmera conectada à Raspberry Pi (normalmente uma webcam USB) e transmitir essas imagens pela rede, seja por cabo Ethernet ou por WiFi. Uma das finalidades para este sistema é a de poder usar a Raspberry Pi com uma câmera como um sistema de vigilância. Este sistema não possui interface gráfica acessível pela sua saída de vídeo, sendo utilizável diretamente apenas por linhas de comando. Porém também é possível acessar a várias ferramentas do sistema através do navegador um computador conectado na mesma rede em que se encontra a Raspberry Pi com o motionEyeOS. Ao acessar a página do sistema com o IP da Raspberry Pi com motionEyeOS (Fig. 2.15) pelo navegador, são encontradas diversas opções como a resolução e a taxa de frames da câmera, opções de streaming, detecção de movimentos, entre outros.

O principal motivo de se utilizar este sistema operacional neste trabalho é a sua leveza. Por ser um sistema que exige pouca capacidade de processamento, ele permite que a Raspberry Pi utilize mais recursos para a transmissão das imagens da câmera. Este sistema também possui desvantagens, como não ser capaz de instalar nada nele. Para adicionar alguma coisa no sistema, é necessário compilá-lo com as novas ferramentas e instalar o novo sistema compilado na memória da Raspberry Pi.

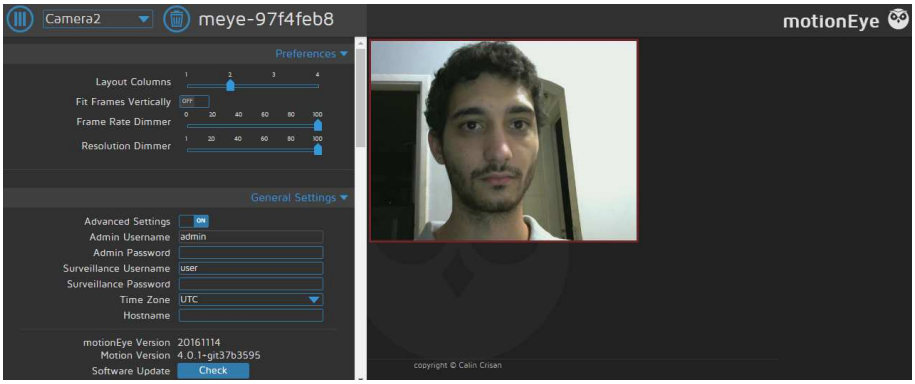


Figure 2.15: Interface do motionEyeOS quando acessado por seu IP através de um navegador no computador

2.3.6 Trinus VR

Trinus VR[22] (Fig. 2.16) é um aplicativo que funciona no sistema operacional Android[5] e um programa que funciona no sistema operacional Windows. A principal função do programa executado em Windows é transmitir as imagens de uma janela por rede WiFi ou cabo USB para o smartphone. O papel do aplicativo é de receber as imagens transmitidas pelo programa rodando em Windows, com a opção de serem mostradas em um formato estereoscópico, ou seja, uma imagem para cada olho, da mesma forma que em qualquer dispositivo de realidade virtual. Ele também possui algumas opções como o rastreamento da orientação da cabeça para controlar o mouse e o ajuste do tamanho da imagem para adaptar ao tipo de HMD utilizado.

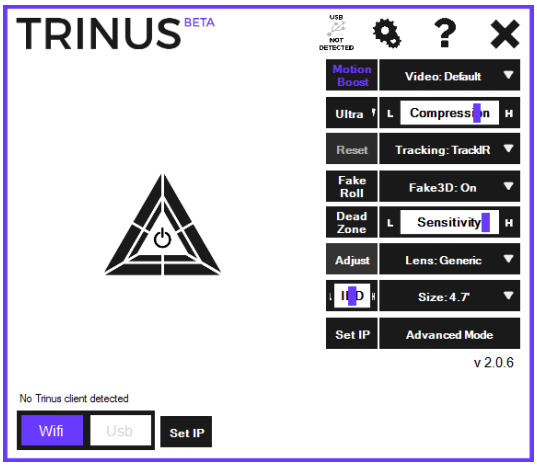


Figure 2.16: Interface do programa Trinus VR executado em Windows

[5]Trinus VR Lite: o aplicativo pode ser obtido em https://play.google.com/store/apps/details?id=com.loxai.trinus.test. Acesso em: 08 dez. 2016

Figure 2.17: Smartphone executando o aplicativo Trinus VR com a imagem no formato de realidade virtual

2.3.7 IMU+GPS-Stream

IMU+GPS-Stream (Fig. 2.18) é um aplicativo que funciona no sistema operacional Android[6]. Sua principal função é a de capturar e transmitir por rede WiFi em UDP (User Datagram Protocol) os dados dos diversos sensores contidos no smartphone. Transmitir por UDP[23] significa que as informações são enviadas sem o emissor se preocupar se receptor dessas informações está recebendo elas corretamente, o que diminui a quantidade de dados a ser transmitida e o tempo para que as informações sejam recebidas. No aplicativo existe a opção de selecionar para qual endereço IP e porta os pacotes serão enviados, além da frequência em que os pacotes são transmitidos. Este aplicativo também é capaz de armazenar em algum arquivo no smartphone os dados capturados.

Alguns dos sensores que o aplicativo é capaz de ler são: acelerômetro, girômetro, magnetômetro, GPS, entre outros. O formato das informações no pacote transmitido pelo aplicativo IMU+GPS-Stream é uma cadeia de caracteres, sendo que os caracteres são sempre números separados por vírgulas e espaços. Os números enviados possuem as informações do tempo em que o pacote é enviado, um identificador para cada sensor um sensor (por exemplo, acelerômetro é o sensor 3) e após cada identificador os dados daquele respectivo sensor (após o 3, vem os valores de aceleração em x, em y e em z respectivamente).

[6]Sensorstream IMU+GPS: o aplicativo pode ser obtido em https://play.google.com/store/apps/details?id=de.lorenz_fenster.sensorstreamgps. Acesso em: 08 dez. 2016

Figure 2.18: Interface do aplicativo IMU+GPS-Stream

3 METODOLOGIA E DESENVOLVIMENTO

São necessários vários processos para que ocorra o funcionamento completo do sistema proposto neste trabalho. Estes processos incluem: capturar os dados dos dispositivos que determinam as posições e ângulos das partes do humano teleoperador, transmitir os dados por algum sistema de comunicação, processar os dados para determinar os movimentos que o robô irá realizar, enviar os comandos de movimentação para o robô, capturar as informações do ambiente em que o robô se encontra, transmitir as informações do ambiente para um dispositivo usado pelo teleoperador e apresentar as informações do ambiente no dispositivo para o teleoperador. Para possibilitar a realização de todos estes processos, os seguintes materiais foram escolhidos para efetuar a sua execução:

- Google Cardboard;
- Microsoft Kinect;
- Bracelete Myo;
- Robô humanoide NAO da Aldebaran Robtics;
- Webcam USB;
- Computador(es) com sistemas operacionais Windows e Ubuntu;
- Raspberry Pi 2 com sistema operacional motionEyeOS;
- Smartphone com sistema operacional Android;
- Roteador com rede WiFi;
- Software para Windows e aplicativo para Android Trinus VR;
- Aplicativo para Android IMU+GPS-Stream;
- Código-fonte em linguagem python rodando em ROS (Robot Operating System).

Abaixo são mostradas duas figuras que mostram: as partes do sistema de teleoperação (Fig. 3.1); e as partes do código-fonte implementado para controlar o robô (Fig. 3.2).

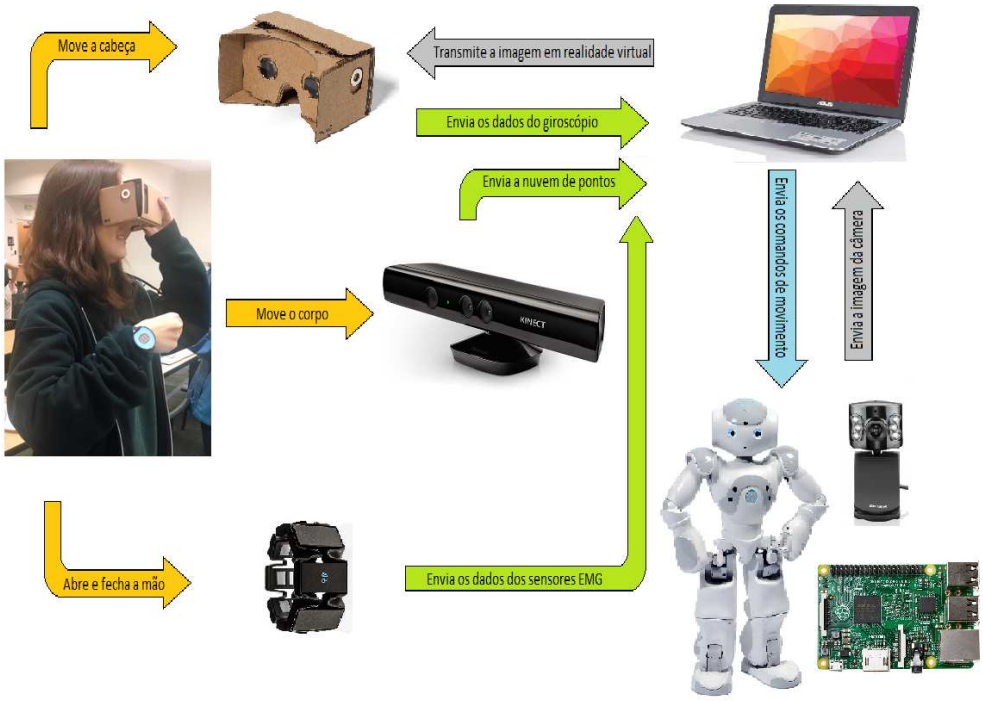


Figure 3.1: Esquema resumindo todo o sistema de teleoperação

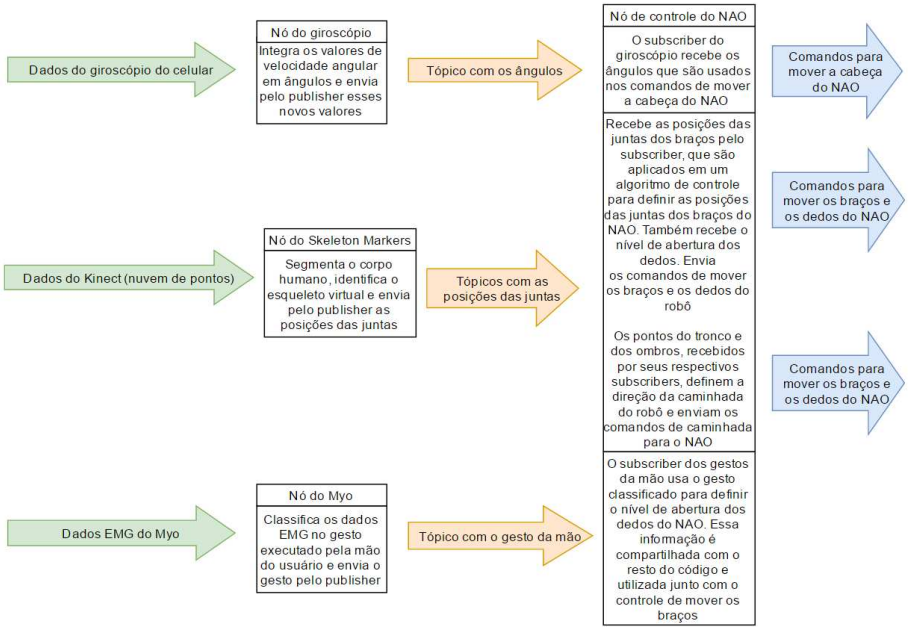


Figure 3.2: Esquema resumindo todo a estrutura do código rodando em ROS

Para a implementação deste trabalho, foi necessário encontrar soluções para dois problemas que resumem os processos descritos acima. Como capturar e transmitir os movimentos do teleoperador para que o robô execute estes movimentos, e como capturar, transmitir e mostrar o que o robô percebe no ambiente ao seu redor para o teleoperador.

Grande parte da solução destes dois problemas reside no smartphone, que possui conexão wireless, uma IMU e, quando em conjunto com o Cardboard (Fig. 2.11), pode ser utilizado como um equipamento de realidade virtual ou de telepresença. Neste caso, o ambiente a ser experimentado pelo teleoperador não será um ambiente virtual, mas sim a imagem da câmera que vem do robô que será visualizada em um formato de realidade virtual. Isso significa que o smartphone é capaz de capturar e transmitir alguns dos movimentos do teleoperador, assim como receber e mostrar as informações do ambiente em que o robô é inserido de forma que o teleoperador se sinta imerso neste ambiente.

3.1 TRANSMISSÃO DOS DADOS DO TELEOPERADOR E CONTROLE DO ROBÔ

3.1.1 Controle da cabeça

Com o smartphone dentro do Cardboard, que é acoplado à cabeça do teleoperador, os movimentos realizados pela cabeça do teleoperador são percebidos pela IMU do smartphone, cujos dados podem ser transmitidos por rede WiFi. Esta é a função do aplicativo IMU+GPS-Stream (Fig. 2.18), obter os dados dos sensores do smartphone em tempo real e transmiti-los por rede WiFi. Estas informações são recebidas pelo computador com sistema operacional Ubuntu rodando o código-fonte em ROS, que recebe os dados da IMU do smartphone pela rede wireless e transforma esses dados nos ângulos para onde as juntas da cabeça (Fig. 3.3) do robô irão se mover.

Os dados transmitidos pelo aplicativo IMU+GPS-Stream são transmitidos na rede WiFi por pacotes utilizando protocolo UDP, então foi implementado um nó no código rodando em ROS responsável por receber essas informações utilizando um socket para ler uma cadeia de caracteres em uma porta específica. Após ser feita a leitura da cadeia de caracteres, é necessário quebrá-la nas diversas informações que ela contém, o que é feito na forma de tokens (pequenas strings que contém os caracteres), sendo que existem caracteres especiais que definem quando separar um token do outro na cadeia (no caso dos pacotes enviados pelo IMU+GPS-Stream, esses caracteres são vírgulas).

Após obter os tokens, é feita a identificação dos dados para saber o tempo de cada pacote, de quais sensores foram lidos os dados e quais são os valores obtidos. Dentre esses valores, está a velocidade angular em radianos por segundo em torno dos eixos X, Y e Z. Com a obtenção do tempo dos pacotes e das velocidades angulares, é possível determinar os ângulos yaw, pitch e roll por integração, assumindo que a velocidade angular é constante entre cada pacote. Esses ângulos são tratados para que eles não ultrapassem os valores máximos que as juntas do pescoço do robô podem assumir.

Após obter os ângulos do smartphone, que também são os ângulos da cabeça do teleoperador, eles são publicados em um tópico que será lido por outro nó. Este outro nó está inscrito no mesmo tópico em que são publicados os ângulos e seu subscriber é responsável por ler esse tópico e obter esses ângulos. Após este último nó obter os ângulos, ele utiliza um proxy do NAOqi de um módulo de movimento ALMotion, que possui um método setAngles cuja função é mover as juntas do NAO para um valor específico, sendo utilizados neste método os valores dos ângulos obtidos. O computador com Ubuntu e o robô NAO (Fig. 2.3) também estão conectados em rede, o que permite que as informações de movimento que o robô irá reproduzir sejam transmitidas do computador para o robô.

Também utilizando o mesmo proxy, é utilizado outro método , getAngles do módulo ALMotion, para obter os ângulos das juntas, sendo feito um comando e uma leitura para cada iteração do nó. Com a leitura dos ângulos pelos sensores das juntas utilizando o método do proxy, são gerados os resultados que serão mostrados no próximo capítulo.

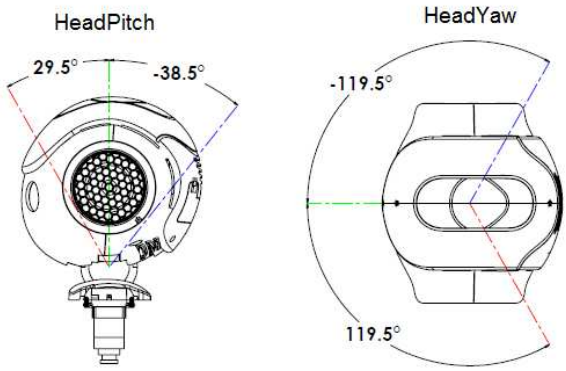


Figure 3.3: Cabeça do NAO e suas juntas (Fonte: Aldebaran)

3.1.2 Controle dos braços e da caminhada

Apenas a movimentação da cabeça, capturada pela IMU do smartphone, é muito pouco para a reprodução dos movimentos de todo o corpo humano, além de não permitir a locomoção no ambiente e nem de interagir com os objetos do ambiente. Então, para a movimentação das outras partes do corpo foi utilizado o Kinect (Fig. 2.4) conectado ao computador com Ubuntu, que é capaz de capturar o movimento de todo o corpo humano e reproduzi-lo em um esqueleto virtual utilizando um pacote do ROS chamado chamado de skeleton_markers (Fig. 3.4).

O pacote openni_tracker, que é utilizado pelo skeleton_markers, determina as posições espaciais (em X, Y e Z) das juntas do esqueleto virtual. O skeleton_markers possui um nó que publica essas posições das juntas em um tópico e outro nó as recebe por um subscriber. Este outro nó foi uma das implementações de código aproveitadas pelo trabalho de Carvalho[1]. Este nó foi modificado neste trabalho para executar mais algumas tarefas: controlar a caminhada do robô, o abrir e fechar dos dedos do robô, além de incluir o controle da cabeça do robô descrito acima. Neste nó é aplicado um algoritmo de controle utilizando cinemática inversa para determinar os ângulos para onde as juntas dos braços do robô irão se mover baseado nas posições das juntas dos braços do esqueleto virtual. Com as posições das juntas calculadas, é utilizado o proxy para enviar os comandos de mover as juntas do robô para os ângulos determinados com a função setAngles do módulo ALMotion.

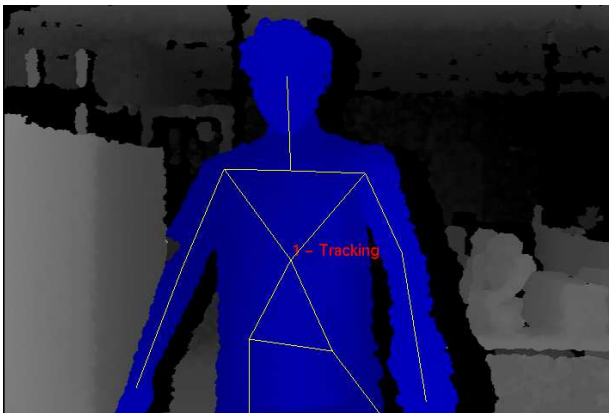
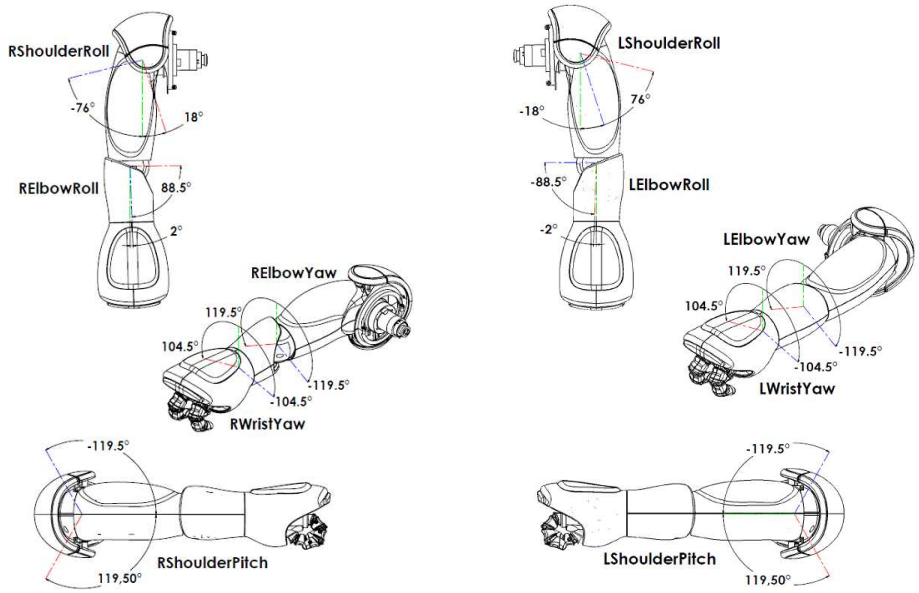


Figure 3.4: Visualização do esqueleto virtual gerado pelo skeleton_markers, desenvolvido para o ROS

Cada ponto do esqueleto, que se refere às juntas entre dois segmentos, possui seus próprios valores em X, Y e Z no espaço e é possível saber a posição relativa de um ponto até o outro, assim como a orientação. Através da orientação entre dois pontos é possível determinar o ângulo para onde as juntas que o robô devem se mover. As transformações de posição e orientação de um ponto até o outro são feitas utilizando quatérnios duais. Isso é utilizado para descobrir os ângulos das juntas para onde os braços do NAO (Fig. 3.5) devem se mover, sabendo a posição e orientação relativas dos pontos de cada cotovelo aos respectivos pontos dos ombros e dos pulsos aos respectivos cotovelos. Os ângulos determinados são usados para mover as juntas dos ombros (R/LShoulderRoll e R/LShoulderPitch) e dos cotovelos (R/LElbowRoll e R/LElbowYaw) do robô.

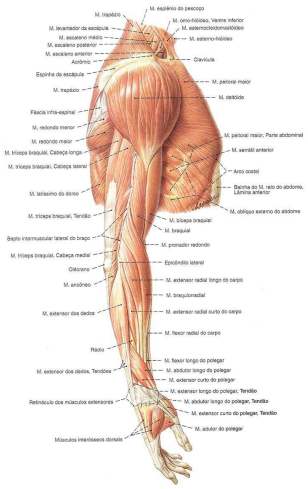
Vale ressaltar que parte do código-fonte, incluindo a implementação da movimentação dos braços com o Kinect, assim como o uso de quatérnios duais, foi completamente aproveitada do trabalho de Cavalho[1]. No trabalho de Carvalho, foi desenvolvido o controle de trajetória dos membros com quatérnios duais e Jacobianas (relação entre os ângulos das juntas e as posições dos membros), de forma que são calculados os ângulos para onde as juntas dos ombros e dos cotovelos devem se mover de acordo com a posição atual dos membros e da posição desejada vinda dos dados do Kinect.



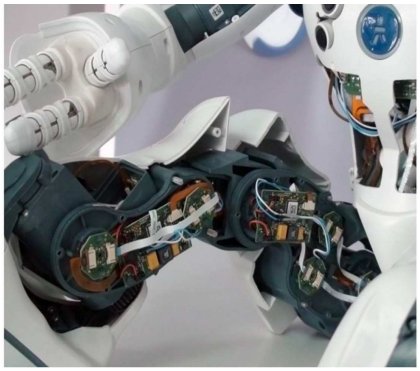
(a) (b)

Figure 3.5: Braços do NAO com suas respectivas juntas (Fonte: Aldebaran)

Enquanto em cada braço do robô existem apenas quatro graus de liberdade, dois de rotação no ombro e dois de rotação no cotovelo, um ser humano é capaz de executar movimentos muito mais complexos com seus braços devido à sua anatomia. Em [24], são definidos os vários movimentos possíveis que um ser humano pode fazer, incluindo as articulações do ombro e do cotovelo no membro superior. Entre esses movimentos estão incluídos flexão, extensão, abdução, adução e vários outros. Dessa forma é possível perceber as diferenças entre o braço humano e o braço robótico do NAO (Fig. 3.6), devido à estrutura muscular e óssea do braço humano, que é muito diferente do acoplamento de peças e engrenagens. Isso significa que não é possível representar todos os movimentos de um braço humano simplesmente pela rotação ao redor de alguns eixos, como é feito com o braço do NAO.



(a) Estrutura muscular de um braço humano (Fonte: Anatomia Online)



(b) Estrutura interna do braço do NAO (Fonte: RoboSavvy)

Figure 3.6: Comparação da estrutura do braço humano com o braço do NAO

Foi tomada a decisão de não fazer o controle das pernas robô obedecendo aos movimentos das pernas do teleoperador. As justificativas para esse método não ter sido implementado são a dificuldade em implementar uma caminhada estável para que o robô não caia e que o teleoperador não é capaz de enxergar por onde caminha com seu corpo enquanto utiliza o HMD. Foi implementada a alternativa de utilizar a inclinação do tronco e a rotação dos ombros para comandar a caminhada do robô.

Também no esqueleto virtual do skeleton_markers, foi utilizada a posição relativa do ponto do pescoço em relação ao ponto do tórax no eixo Z (profundidade). A partir disso, é possível saber se o teleoperador está inclinando o corpo para frente ou para trás, o que ativa o parâmetro da caminhada do robô em linha reta para frente ou para trás. Este parâmetro é uma taxa da velocidade máxima que o robô é capaz de andar utilizando a função moveToward. O parâmetro é definido como -0.8 para andar para trás, 0 para ficar parado e 0.8 para andar para frente. Também foi utilizada a posição relativa do ponto do ombro direito em relação ao ponto do ombo esquerdo no eixo Z. Dessa forma, é possível determinar se o teleoperador está girando o corpo para esquerda ou para direita, o que ativa o parâmetro da caminhada do robô girando no sentido horário ou anti-horário em torno de si. Este parâmetro, que é uma taxa da velocidade da caminhada girando, é definido como -0.8 para andar girando para a direita (sentido horário), 0 para ficar parado e 0.8 para andar girando para a esquerda (sentido anti-horário).

A tomada de decisão de se utilizar apenas os parâmetros 0, 0.8 ou -0.8 se devem à simplicidade nos comandos para que ocorra uma resposta mais previsível da caminhada com poucos estados possíveis. O robô pode apenas ficar parado, carminhar para frente ou para trás com velocidade constante, ou girar nos sentidos horário ou anti-horário com velocidade angular constante. A taxa 0.8 escolhida se mostrou boa o suficiente para que o robô não caminhasse muito lento ao se locomover no ambiente e nem muito rápido para que ocorram muitas perdas no equilíbrio e mantenha a facilidade no controle da posição do robô.

É possível combinar os dois parâmetros de caminhada se o teleoperador inclinar e girar o seu tronco, o que fará com que o robô caminhe em uma trajetória circular. Após a descoberta de ambos os parâmetros, é utilizado o proxy com os parâmetros definidos da caminhada do robô na função moveToward do módulo ALMotion. A função moveToward utiliza como argumentos de entrada as velocidades relativas em X (frente e trás), Y (para os lados) e theta (girar no sentido horário ou anti-horário), sendo que cada um desses valores varia de -1 a 1. Aqui são definidas as velocidades em X e theta, sendo que Y é sempre usado como 0. Também é utilizado o proxy para obter a velocidade da caminhada no eixo X (para frente e para trás) e a velocidade angular de caminhada com a função getRobotVelocity do módulo ALMotion.

A caminhada executada pelo robô ao chamar a função moveToward utiliza um modelo de pêndulo invertido linear, além de ser feito todo o planejamento de trajetória para cada passo que o robô executa (onde o robô irá pisar) e o controle para estabilizar a caminhada na presença de perturbações. Mais informações a respeito dessa caminhada podem ser encontradas em [25, 26].

3.1.3 Controle dos dedos da mão

Para uma maior interação do usuário através do robô com o ambiente, foi escolhido implementar a transmissão dos gestos capturados pelo Myo para que o robô possa abrir e fechar os dedos (Fig. 4.6). Dessa forma, é possível utilizar o robô para pegar objetos com as mãos e carregá-los consigo, fornecendo ao usuário uma opção para interagir com os objetos do ambiente usando o corpo do robô.

A implementação da ação de abrir e fechar os dedos do robô é feita com apenas 3 posições: aberto, folgado e fechado. Foi utilizado o pacote ros_myo que possui um classificador pronto que, a partir dos dados dos sensores EMG, é capaz de decidir qual dessas 3 posições o teleoperador está executando no momento. O gesto classificado é enviado pelo publisher do nó do ros_myo e recebido pelo subscriber do mesmo nó que controla a cabeça, os braços e a caminhada do robô. Dentro do nó que possui o subscriber, é utilizado o proxy para comandar o quanto o robô deve abrir os dedos dependendo do gesto obtido utilizando o método setAngles do módulo ALMotion. Também é utilizado o proxy para obter o nível de abertura dos dedos com o método getAngles do módulo ALMotion.

3.2 TRANSMISSÃO DAS IMAGENS DA CÂMERA E APRESENTAÇÃO NO FORMATO DE REALIDADE VIRTUAL

O método utilizado para perceber o ambiente ao redor do robô é a imagem da câmera. O robô NAO possui duas câmeras em sua cabeça, mas também é possível acoplar uma webcam em sua cabeça. Foram testados vários métodos para transmitir as imagens da câmera no robô até o smartphone no Cardboard usado como HMD. Os métodos testados foram: a câmera superior do NAO, uma webcam USB conectada ao computador, uma câmera IP (além da câmera, possui a parte de WiFi integrada) e uma webcam USB conectada à Raspberry Pi. Nos resultados são mostrados os detalhes das transmissões de cada método.

Dentre os vários métodos testados, o mais simples é a utilização da câmera superior do próprio robô, por não ser necessário acoplar um objeto ao robô. Este método não foi utilizado pela baixa taxa de frames e resolução obtidas, sendo necessário sacrificar muito um dos dois fatores para conseguir o outro a um nível satisfatório para fornecer ao usuário uma experiência de imersão agradável, além de ser necessário a utilização de um cabo ethernet.

O segundo método testado foi com o uso de uma webcam USB, que forneceu os melhores resultados em termos de qualidade da imagem por resolução e taxa de frames. Este método não foi utilizado porque o comprimento do cabo que conecta a webcam ao computador é muito curto e não fornece ao robô liberdade para se locomover.

O terceiro método testado foi com o uso de uma câmera IP, que fornece resultados muito semelhantes aos da webcam USB em termos de resolução e taxa de frames. Este método não foi utilizado devido à instabilidade na transmissão das imagens. Em grande parte dos testes, as imagens deixavam de ser transmitidas pouco tempo depois do início da recepção delas no computador, sendo necesário atualizar a página no navegador que acessa a câmera ou reiniciar a câmera sempre que isso ocorre.

Por motivos de desempenho e qualidade da imagem, o método escolhido como melhor foi utilizar

uma Raspberry Pi 2 (Fig. 2.9) com sistema operacional motionEyeOS, alimentada por uma bateria e ligada a uma webcam USB acoplados ao robô. Esta solução oferece ao usuário uma solução com imagens nítidas e fluídas, com relativamente baixos tempos de resposta, além de maior mobilidade ao robô para a locomoção. Normalmente se utiliza o Raspbian, um sistema operacional muito parecido com o Debian feito para a Raspberry Pi, porém a captura e transmissão das imagens da câmera neste sistema operacional sobrecarregam o processador da Raspberry Pi e reduzem muito a taxa de frames.

O motionEyeOS é focado em simplesmente capturar as imagens da webcam USB e transmiti-las pela rede. Este modelo de Raspberry Pi não possui WiFi embutida, então para utilizar este recurso é necessário instalar um adaptador WiFi USB. Como alternativa à conexão por WiFi, pode ser utilizada a conexão por cabo Ethernet. A imagem da câmera é transmitida em tempo real para o computador com sistema operacional Windows. É necessário utilizar um segundo computador ou uma máquina virtual com outro sistema operacional além do Ubuntu devido à limitação do software Trinus VR. Com a visualização das imagens da câmera no computador com Windows, o Trinus VR é responsável por transmiti-las pela rede para o smartphone, por onde o teleoperador irá visualizá-las no formato de realidade virtual.

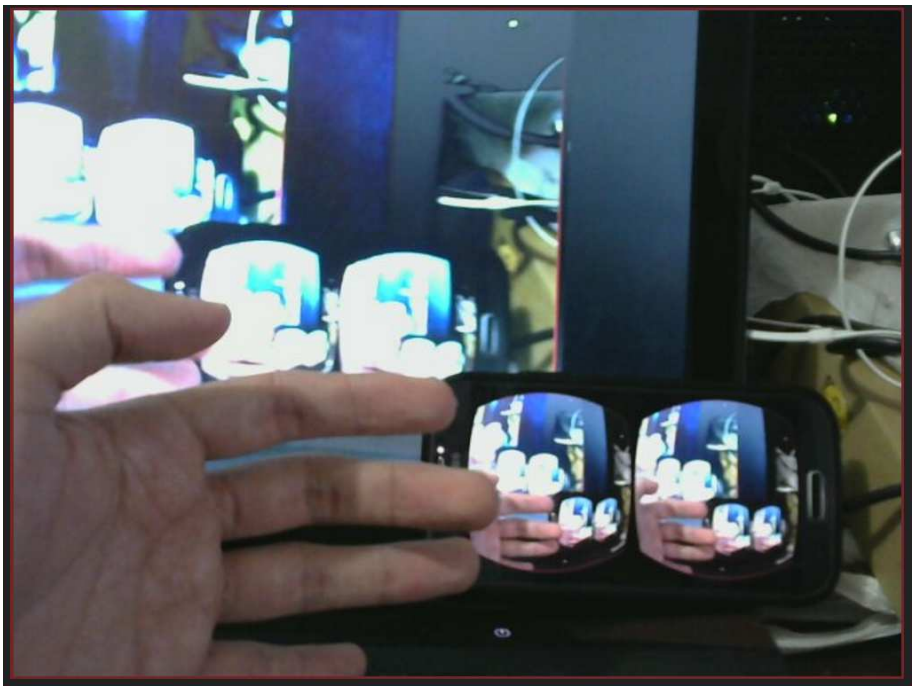


Figure 3.7: Imagem da câmera capturando a tela do computador e a tela do smartphone no formato de realidade virtual

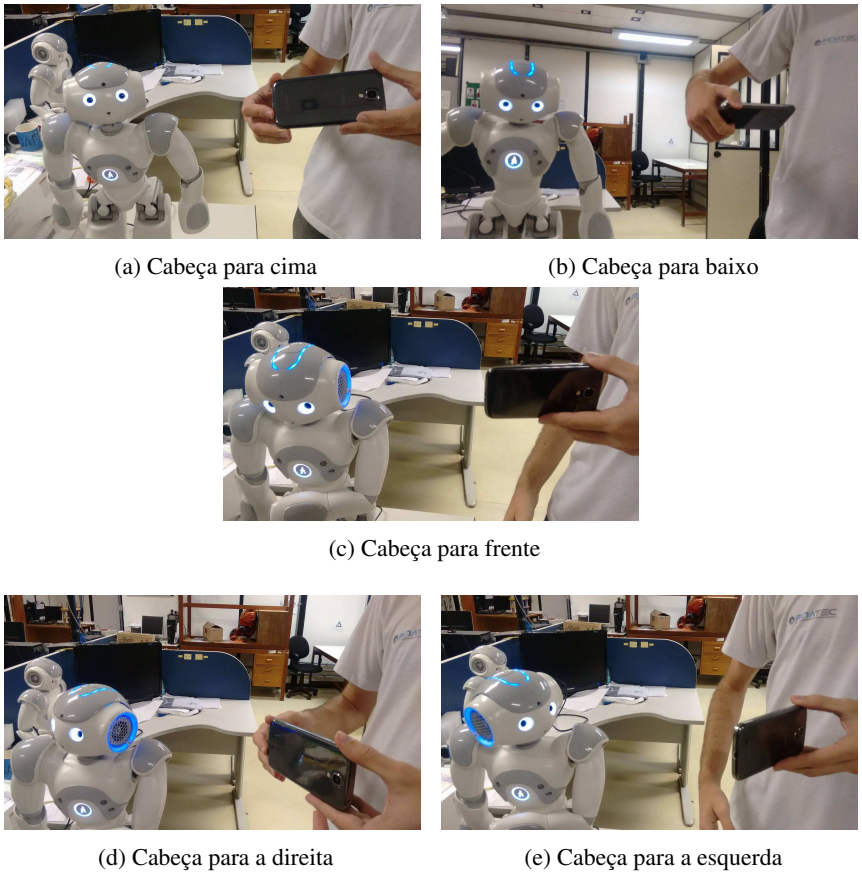
Na figura acima é possível perceber que a imagem capturada pela câmera, que é transmitida pelo motionEyeOS pela rede, é reproduzida no computador e no smartphone em formato de realidade virtual com o uso do Trinus VR.

4 RESULTADOS

A maioria dos equipamentos utilizados possuem limitações que afetam na qualidade dos resultados obtidos durante a implementação deste trabalho, principalmente considerando que é necessário que vários equipamentos funcionem como previsto simultaneamente. Durante a implementação, foi possível obter um resultado que faz com que o robô se mova seguindo os movimentos do corpo do teleoperador com os movimentos programados e também a imagem da câmera seja obtida no formato de realidade virtual no smartphone sendo mostrada para o usuário.

4.1 CONTROLE DA CABEÇA

Foi testado o controle da cabeça girando o smartphone para cima, para baixo, para a esquerda e para a direita e verificando a resposta com os movimentos da cabeça do robô (Fig. 4.1).



(a) Cabeça para cima

(b) Cabeça para baixo

(c) Cabeça para frente

(d) Cabeça para a direita

(e) Cabeça para a esquerda

Figure 4.1: Robô movendo a cabeça com resposta à rotação do smartphone

Foi possível notar que o robô é capaz de obedecer aos comandos de movimentação da cabeça pro-

gramados. Ao girar o smartphone para cima, a cabeça do robô gira para cima. Ao girar o smartphone para baixo, a cabeça do robô gira para baixo. Ao girar o smartphone para a, a cabeça do robô gira para cima. Ao retornar o smartphone para a posição central, a cabeça do robô é direcionada para frente. Ao girar o smartphone para a direita, a cabeça do robô gira para a esquerda. Ao girar o smartphone para a direita, a cabeça do robô gira para a direita.

Para obter os dados responsáveis pelo controle da cabeça, os dados que são enviados pelo smartphone, pelo publisher do ângulo e pelo nó que envia o comando das juntas da cabeça são armazenados em arquivo, além dos ângulos lidos das juntas da cabeça do robô e dos tempos em que os dados são registrados enquanto o código do sistema de teleoperação funciona. Todos os dados são plotados em um gráfico para que sejam feitas as análises qualitativa e quantitativa dos resultados obtidos.

Os dados dos sensores possíveis de se obter da IMU do smartphone para a movimentação da cabeça são do girômetro e da orientação (que é um cálculo de ângulo derivado do giroscópio, não é realmente um sensor). Ao utilizar a orientação, que fornece diretamente o ângulo, algumas vezes se via a cabeça do robô em posições semelhantes às do teleoperador e às vezes não, em alguns casos chegando a extrapolar o limite do ângulo de rotação do pescoço do robô. Ao utilizar a integração da velocidade angular fornecida pelo giroscópio, foram obtidos resultados mais satisfatórios, mas observou-se um leve movimento da cabeça do robô mesmo quando a cabeça do teleoperador estava parada. Além disso, era muito notável que, apesar de o robô obedecer aos comandos de girar a cabeça, aparecia uma defasagem entre a posição da cabeça do teleoperador e do robô mesmo pouco tempo após o início da execução do sistema de teleoperação.

O tempo de resposta entre a execução de um movimento da cabeça do teleoperador e do robô replicar este movimento foi considerado aceitável, com os valores mais altos por volta de 0,5 segundo e a diferença de ângulo entre os ângulos vindos do giroscópio do celular e do robô no mesmo tempo tiveram os valores mais altos por volta de 20°, como pode ser visualizado nos gráficos abaixo.

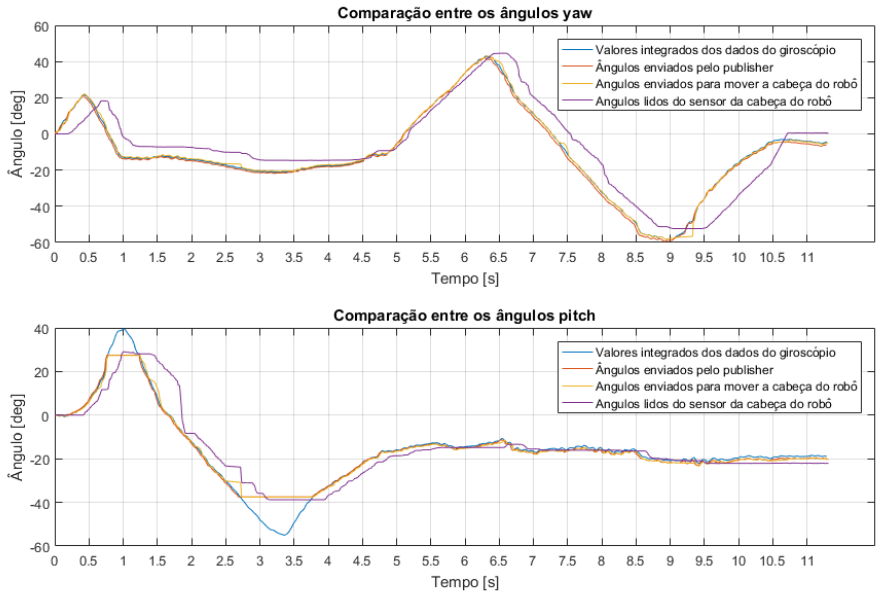


Figure 4.2: Gráfico com os ângulos yaw e pitch a partir dos dados do giroscópio do celular e obtidos da cabeça do robô

Este gráfico não mostra o ângulo real da cabeça do teleoperador. As melhores estimativas desses valores vêm dos ângulos calculados pela integração dos valores vindos do giroscópio do celular (curva azul) e dos ângulos enviados pelo publisher no nó que recebe os valores do giroscópio (curva laranja), que teoricamente deveriam ser os mesmos, mas mostraram uma diferença muito pequena. A maior diferença entre essas duas curvas ocorreu quando os valores integrados eram maiores em módulo que os ângulos máximos que as juntas da cabeça do robô podem se posicionar. Foi notada uma diferença levemente maior entre as duas primeiras curvas e a terceira curva que corresponde aos valores enviados ao robô pelo nó que possui o subscriber (curva amarela).

A quantidade de amostras do nó do publisher no mesmo período de tempo foi maior que a quantidade de amostras do nó do subscriber, isso significa que o segundo nó não foi capaz de processar todas as informações enviadas pelo publisher, já que cada nó gera uma amostra para cada iteração. A última curva se refere aos ângulos obtidos pelos sensores das juntas da cabeça do robô (curva roxa), e é fácil de perceber que existe um atraso entre ela e todas as outras três curvas na maior parte do tempo. Parte desse atraso deve ser causado não apenas pela resposta ao movimento, mas também pelo tempo que a informação dos sensores leva para chegar de volta do robô até o computador pela rede WiFi. Como o giroscópio do celular utilizado não é um dispositivo perfeito, também existem erros entre os ângulos calculados pelos valores vindos do giroscópio e o ângulo real da cabeça do teleoperador.

4.2 CONTROLE DOS BRAÇOS E DA CAMINHADA

Com relação ao movimento dos braços com o Kinect, esta parte não foi implementada neste trabalho, e sim aproveitada por um trabalho feito anteriormente[1]. Foi testado os controle dos braços do robô com o teleoperador movendo os seus braços e verificando a resposta com os movimentos dos braços do robô (Fig. 4.3).

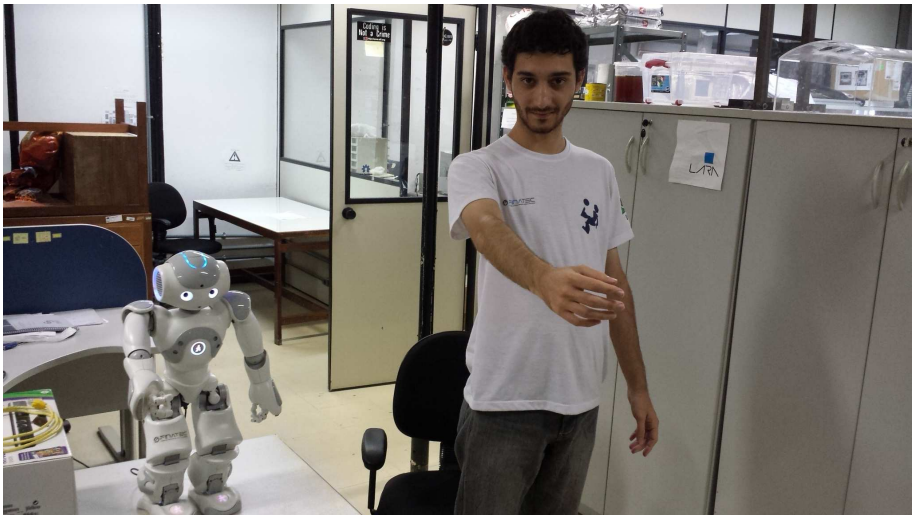


Figure 4.3: Robô respondendo ao movimento do braço do teleoperador

Os movimentos dos braços muitas vezes ocorriam como esperado, imitando o movimento dos braços do teleoperador, mas o robô não foi capaz de reproduzir os movimentos dependendo da tra-

jetória que o teleoperador utilizava para ir de uma posição até a outra. O robô não foi capaz de reproduzir alguns movimentos como levantar completamente o braço e dobrar o cotovelo para trás. De certa forma isso é esperado porque o corpo humano é diferente do corpo do robô humanoide, apesar de ser semelhante, então não é possível reproduzir perfeitamente os movimentos de algumas juntas.

Para obter os dados responsáveis pelo controle da caminhada, as taxas das velocidades utilizadas pelo nó que envia o comando de caminhada são armazenadas em arquivo, além das velocidades lidas do robô e dos tempos em que essas velocidades são registradas durante a execução do código do sistema de teleoperação. Todos os dados são plotados em um gráfico para que sejam feitas as análises qualitativa e quantitativa dos resultados obtidos.

Foi testado os controle da caminhada do robô com o teleoperador inclinando o seu tronco e girando os seus ombros e verificando a resposta com a caminhada do robô (Fig. 4.4).

(a) Caminhada para frente (b) Caminhada para trás

(c) Caminhada girando para a direita (d) Caminhada girando para a esquerda

Figure 4.4: Robô executando a caminhada de acordo com a posição do tronco e dos ombros do teleoperador

Foi possível notar que o robô é capaz de obedecer aos comandos de caminhada programados. Quando o teleoperador inclina o corpo para frente, o robô anda para frente. Quando o teleoperador inclina o corpo para trás, o robô anda para trás. Quando o teleoperador gira os ombros para a esquerda, o robô anda girando para a esquerda. Quando o teleoperador gira os ombros para a direita, o robô anda girando para a direita.

Se tratando dos movimentos de caminhada, não foram utilizadas réplicas dos movimentos das pernas com o Kinect por ser algo muito difícil de implementar, principalmente considerando a questão de manter o equilíbrio do robô. Foi implementada a aquisição de posições do tronco para definir os parâmetros de um algoritmo de caminhada já implementado. A inclinação do tronco do operador define se o robô anda para frente e para trás e a rotação do tronco (diferença da posição entre os ombros) define se o robô gira no sentido horário ou anti-horário. Com isso foi possível controlar a caminhada do robô dentro do esperado utilizando os movimentos do corpo do operador.

Os problemas encontrados quando o teleoperador faz o robô caminhar são principalmente o tempo de resposta para ativar o movimento. Normalmente leva mais de um segundo entre o tempo em que o teleoperador inclina o seu corpo e o NAO começa a andar. Abaixo é ilustrado um gráfico comparando a taxa da velocidade máxima de caminhada enviada (que pode variar de -1 a 1) com a velocidade executada pelo robô para caminhar na direção X (para frente ou para trás), em metros por segundo, e girar (sentido horário ou anti-horário), em radianos por segundo.

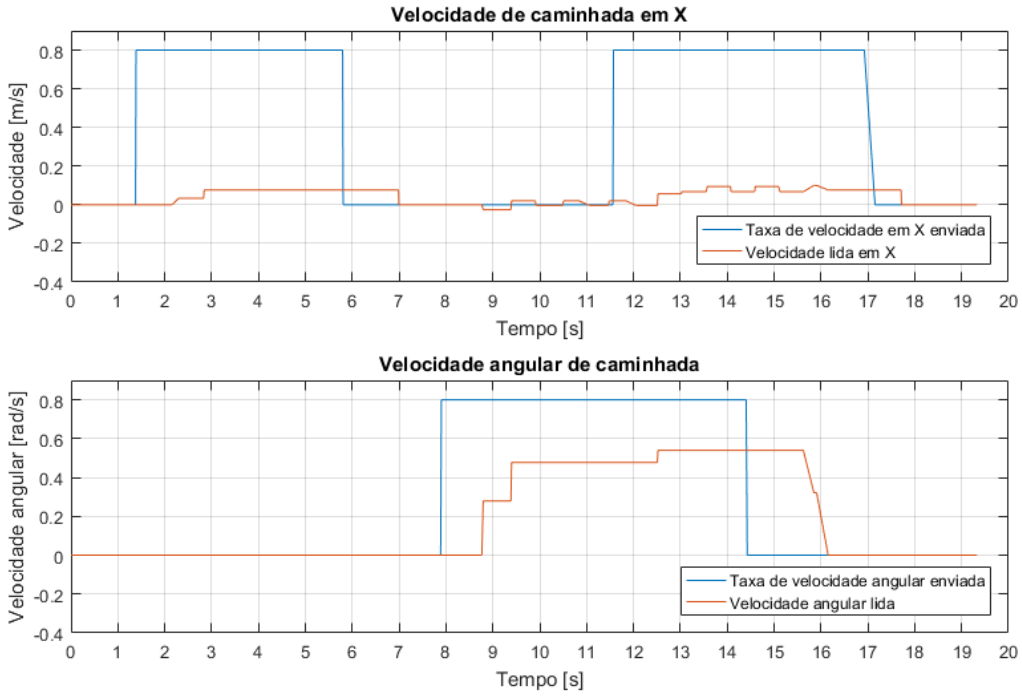


Figure 4.5: Gráfico com as taxas das velocidades enviadas e as velocidades executadas e lidas

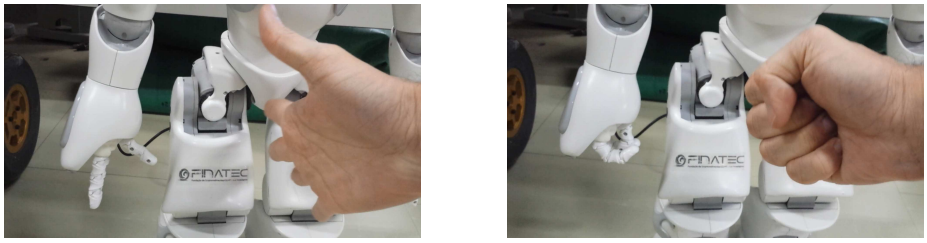
Nos gráficos acima, as curvas azuis representam as taxas da velocidade máxima enviada e as curvas laranjas representam as velocidades na direção X ou angulares executadas pelo robô. É fácil

perceber o atraso na resposta do comando acionado pelo teleoperador inclinando e girando o corpo até que o robô responda iniciando ou parando o movimento. Pelo que é possível identificar no gráfico, o tempo de resposta entre o envio do comando e o início da sua execução normalmente é por volta de 1 s.

4.3 CONTROLE DOS DEDOS DAS MÃOS

Para obter os dados responsáveis pelo controle da abertura e fechamento dos dedos, são armazenados em arquivo o nível de abertura dos dedos utilizado pelo nó para enviar o comando e os dados lidos da abertura dos dedos, além dos tempos em que os dados são registrados enquanto o código do sistema de teleoperação funciona. Todos os dados são plotados em um gráfico para que sejam feitas as análises qualitativa e quantitativa dos resultados obtidos.

Foi testado os controle dos dedos das mãos do robô com o teleoperador abrindo e fechando a mão cujo braço estava com o Myo e verificando a resposta com o abrir e fechar dos dedos do robô (Fig. 4.6).



(a) Dedos abertos (b) Dedos fechados

Figure 4.6: Dedos do NAO respondendo ao comando de abrir e fechar os dedos

Foi possível notar que o robô é capaz de obedecer aos comandos de abertura e fechamento dos dedos programados. Quando o teleoperador abre a sua mão, o robô abre os seus dedos. Quando o teleoperador fecha a sua mão, o robô fecha os seus dedos.

Com respeito ao abrir e fechar dos dedos com o uso dos sensores EMG do Myo, o robô normalmente sempre responde aos comandos como esperado, porém ele normalmente leva um tempo pequeno até responder ao movimento do teleoperador. No gráfico abaixo é possível ter uma noção sobre o tempo de resposta entre o envio do comando e quando os dedos do robô respondem ao comando.

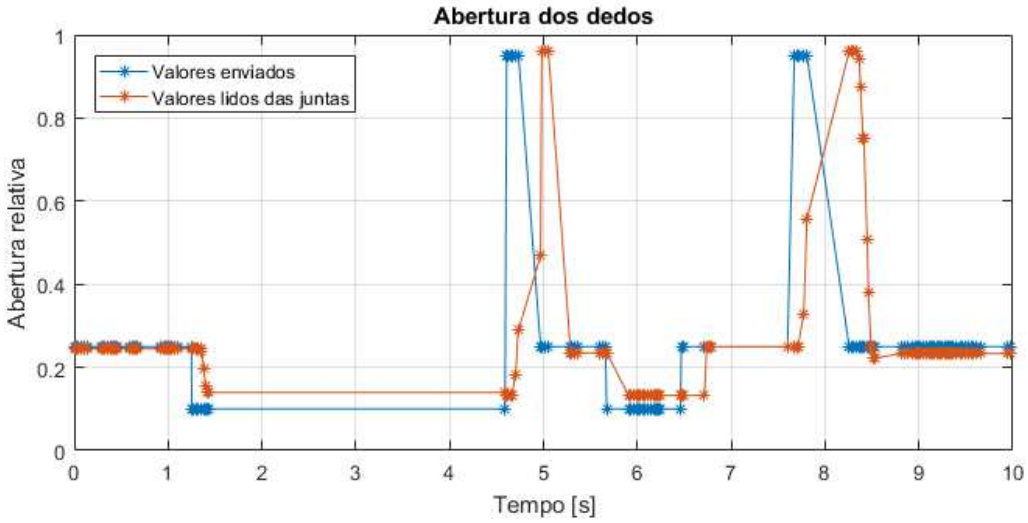


Figure 4.7: Gráfico com a abertura relativa dos dedos enviada e obtida a partir do sensor do robô

No gráfico acima, 1 seria a abertura máxima dos dedos e 0 significaria o fechamento completo dos dedos. A primeira curva se refere aos valores enviados para o robô (curva azul) depois de classificar os dados dos sensores EMG, decidindo qual dos 3 gestos o teleoperador está executando com a mão e definindo a abertura dos dedos. A segunda curva se refere aos valores lidos do sensor dos dedos (curva laranja) indicando qual o nível de abertura deles. Os símbolos com formato de asteriscos indicam o pontos em que ocorreram a amostragem.

É possível perceber que alguns pontos estão muito próximos uns dos outros, mas também existem pontos que possuem uma grande distância temporal entre o seu próximo vizinho. Como cada amostra é gerada a partir de uma iteração do nó, mais uma vez percebe-se que existe uma irregularidade na frequência em que o nó é executado, assim como foi visto nos resultados do controle da cabeça. Além disso, semelhante ao visto nos resultados do controle da cabeça, fica bem clara a existência do atraso à resposta do comando para o valor esperado, sendo algo em torno de 0,5 segundo desde quando o comando é enviado ao robô e o os dedos do robô atingem a posição descrita pelo comando.

4.4 TRANSMISSÃO DAS IMAGENS DA CÂMERA

Os resultados utilizados para avaliar a transmissão das imagens da câmera foram: se as imagens da câmera de fato conseguiam chegar até o smartphone em formato de realidade virtual e a resolução e a taxa em que as imagens eram transmitidas. É considerado que uma taxa a partir de 24 frames por segundo[27] gere imagens relativamente fluidas, que forneçam ao usuário um conforto visual ao visualizar o movimento dos objetos através das várias imagens exibidas. Também é considerado que uma resolução a partir de 640x480, para cada olho, fornece nitidez o suficiente para que o teleoperador seja capaz de visualizar bem o ambiente.

A respeito da transmissão das imagens para a tela do smartphone, pode-se dizer que esta é a parte do trabalho que mais gerou dificuldades até atingir um nível satisfatório, principalmente porque são realizadas até duas transmissões de imagens por rede e pelo menos uma delas é sempre feita por WiFi.

Uma das transmissões é feita da câmera no robô para o computador e a outra é feita do computador para o smartphone com o usuário através do programa Trinus VR em formato de realidade virtual. A transmissão do computador para o smartphone gerou poucos problemas devido ao uso da rede WiFi na banda de 5 GHz, que garante ao usuário liberdade de movimentação por não estar preso a nenhum cabo e uma boa qualidade de transmissão visto pela resolução e taxa de frames obtidas. Os grandes problemas vieram da transmissão da câmera no robô para o computador. Além disso, em todos os casos existe um pequeno atraso na exibição das imagens da câmera para o teleoperador por conta das duas transmissões.

Inicialmente foi testado o uso da transmissão da câmera do robô, mas não foi possível obter uma transmissão com uma taxa de frames e resolução aceitáveis. Para conseguir uma transmissão relativamente aceitável através de cabo Ethernet, foi usada uma resolução muito baixa (320x240 pixels) e a taxa de frames era por volta de 15 frames por segundo (fps), o que gera um certo desconforto na experiência do usuário ao utilizar o sistema de teleoperação. Na mesma resolução, utilizando WiFi, a taxa de frames cai muito, para abaixo de 10 fps, tornando o sistema praticamente inutilizável, já que demora muito para o usuário perceber que algo acontece quando o robô se move respondendo aos movimentos do teleoperador. As possíveis causas para a baixa velocidade de transmissão são: o processamento no envio das informações feitos pelo robô, além do robô ser responsável por processar os comandos dos movimentos das juntas; além da forma utilizada para a aquisição, que foi feita por um script em python fornecido pela Aldebaran, já que outras soluções testadas em Ubuntu que não foram possíveis de ser aplicadas em Windows são capazes de fornecer taxas de transmissão mais altas.

As outras soluções testadas envolvem acoplar uma outra câmera no robô e utilizar algum meio para transmitir as imagens da câmera para o computador e depois para o smartphone no formado de realidade virtual. A segunda opção testada foi o uso de uma webcam USB conectada diretamente ao computador. O computador consegue receber sem problemas as imagens da webcam na resolução máxima deste dispositivo a uma taxa constante de 30 frames por segundo. Foram testadas câmeras com resoluções de 640x480 pixels e 1280x720 pixels. O grande problema de utilizar uma webcam com cabo USB conectada ao computador é a extensão do cabo USB, que na maioria das câmeras é por volta de 1 a 2 metros, o que força o computador ficar próximo do robô e impede que o robô tenha muita liberdade para se locomover.

Na terceira opção testada, foi utilizada uma câmera IP, que, além da câmera em si, possui a própria placa WiFi embutida e transmite a imagem da câmera pela rede. A câmera IP testada possui resolução de 1280x720 pixels e transmite a imagem a 23 fps, o que gerou resultados excelentes considerando o meio de transmissão utilizado. O grande problema da câmera IP se deve à perda muito rápida do sinal de transmissão dependendo do congestionamento da rede no canal utilizado, que sofre interferência de outras redes que estão no mesmo canal. Uma possível solução, para o uso da câmera IP seria utilizar uma outra câmera que suporta a banda WiFi de 5 GHz (que, além de mais estável, possui muito mais canais disponíveis), porém as câmeras IP que suportam essa banda são muito caras (cerca de 150 dólares na amazon.com).

A quarta opção testada, que se tornou a definitiva no momento da escrita deste trabalho, é o uso de uma Raspberry Pi acoplada ao robô com o sistema operacional motionEyeOS. Foi testado utilizar o Raspbian para a transmissão, mas independente dela ser feita por cabo Ethernet, WiFi 2.4 GHz

ou 5 GHz, a transmissão era muito lenta utilizando este sistema operacional porque o processador da Raspberry Pi fica sobrecarregado ao executar essa tarefa. Com o uso do motionEyeOs, como ele possui muito menos recursos e é focado em transmitir as imagens da câmera, foi possível transmitir as imagens da câmera com uma boa resolução (640x480 pixels) com taxas entre 23 fps e 28 fps utilizando cabo Ethernet de 5 metros entre o roteador e a Raspberry Pi, que permite ao robô ter muita liberdade para se locomover.

Também foi testada a transmissão no motionEyeOs por WiFi 2.4 GHz, mas as taxa de frames ainda ficou muito baixa para tornar a experiência do uso do sistema de teleoperação agradável ao usuário. Como o motionEyeOs é muito limitado, não é possível instalar drivers neste sistema, mas é possível compilar todo o sistema e adicionar os drivers nas opções de compilação que o criador do sistema fornece. O motionEyeOS reconheceu o adaptador WiFi que suporta apenas a banda de 2.4 GHz, mas não reconheceu o adaptador que suporta ambas as bandas de 2.4 GHz e 5 GHz, então não foi possível testar o motionEyeOS com transmissão pela banda de 5 GHz, que poderia ter sido a melhor solução para a transmissão da imagem da câmera além do uso de uma câmera IP que suporta a banda de 5 GHz. Com o uso da transmissão por WiFi na banda de 5 GHz, seria possível fornecer uma melhor transmissão das imagens, além da total liberdade para a locomoção do robô já que ele não estaria conectado a nenhum cabo que limite os seus movimentos.

4.5 TESTE COMPLETO

Também foi realizado um exercício com o sistema controlando o robô e posicionando um objeto a uma certa distância da posição inicial do robô, mas com uma certa facilidade para que fosse possível agarrar o objeto com os dedos do robô. Nesse exercício é possível testar completamente a imersão por telepresença, tanto pela visualização do ambiente no formato de realidade virtual, como por ser possível de locomover no ambiente e interagir com algum objeto do ambiente. O robô inicialmente faz uma combinação de movimentos de andar em linha reta para frente ou para trás e de girar para algum dos lados para se posicionar próximo e de frente para o objeto a ser agarrado. Quando chega na posição correta, é feita a movimentação de um dos braços até que uma das mãos esteja bem próxima do objeto e finalmente o objeto possa ser agarrado ao fechar os dedos e ele poder ser carregado com o robô.

(a) Início do teste com o sistema funcionando

(b) Robô caminhando para se aproximar do objeto

(c) Objeto sendo agarrado pelos os dedos do robô

(d) Objeto sendo segurado pela mão do robô

Figure 4.8: Teste do sistema de teleoperação

Devido às características inerentes às partes do sistema, foi possível perceber a dificuldade na execução deeste exercício. O atraso da resposta do robô ao comando de andar para frente ou para trás e a execução imperfeita de alguns dos movimentos dos braços foram os principais motivos para que o exercício levasse mais tempo que o esperado e se tornasse mais difícil. Como o robô não obedecia ao movimento de dobrar o cotovelo para trás, foi necessário colocar o robô em uma posição muito específica depois de aproximar e afastar o robô do objeto várias vezes andando, devido ao atraso na obediência a este comando, para que fosse possível alcançar o objeto com a mão e finalmente agarrá-lo. No fim foi possível completar o exercício, aproximando o robô do objeto e agarrando o objeto com os dedos do robô.

Neste exercício também é avaliada a experiência de imersão e controle do robô do ponto de vista do usuário. Devido à posição utilizada para a câmera, o ponto de vista enxergado pelo teleoperador seria como se os olhos do teleoperador estivessem na cabeça do robô. Devido à altura do robô e da proximidade em que o robô fica dos objetos para que ele possa alcançà-los com as mãos, o teleoperador tem a sensação que, quando imerso na realidade do robô, sua altura é reduzida e o tamanho dos objetos também é aumentada. Apesar de existirem atrasos na transmissão das imagens, eles são pequenos a ponto de serem perceptíveis apenas quando são vistos lado-a-lado (mundo real, tela do computador e tela do smartphone), sendo que este não foi um fator que prejudicou a execução do exercício.

Apesar do controle imperfeito dos braços e dos atrasos no controle dos dedos e da cabeça, o teleoperador ainda consegue mover as partes do seu corpo e visualizar as partes do robô se movmendo de acordo, o que oferece ao teleoperador uma proximidade da sensação de que o corpo do robô é o seu próprio. O fator de imersão é ainda mais ampliado com o fato dos gestos executados pelo teleoperador serem capazes de fazer com que o robô interaja com o ambiente, agarando um objeto no caso deste exercício.

5 CONCLUSÃO E PROPOSTAS

Neste trabalho foi possível implementar com êxito o que foi inicialmente proposto dentro das limitações de tempo, complexidade e tecnológicas esperadas. Foi possível controlar o robô replicando vários dos movimentos do teleoperador, e ativar movimentos programados do robô com parâmetros controlados por outros movimentos do teleoperador sendo capaz de fazer o robô teleoperado se locomover no ambiente e interagir com os objetos do ambiente, além de transmitir as imagens da câmera para o smartphone no formato de realidade virtual para proporcionar a sensação de imersão no ambiente do robô.

Apesar de se ter obtido o que foi proposto e de se ter melhorado muito os resultados obtidos ao longo da execução deste trabalho, ainda foram notadas algumas problemas no processo de implementação deste trabalho que impediram se obter um resultado para que o sistema funcionasse exatamente como era previsto. De certa forma, é esperado que ocorram problemas no processo de implementação do sistema devido ao seu nível de complexidade por envolver vários dispositivos interconectados e funcionando simultaneamente. Uma fator notável no nível de complexidade da implementação deste trabalho é a utilização simultânea de até 5 sistemas operacionais (computador com Windows e Ubuntu, NAO com Gentoo, smartphone com Android e Raspberry Pi com motionEyeOS). Os problemas mais notáveis se referem à transmissão por rede das imagens da câmera (que não foi possível implementar uma solução ótima por WiFi), dos dados da IMU do smartphone, o controle do robô com o Kinect e o tempo de resposta para a execução dos diversos movimentos programados. Possíveis soluções para estes problemas seriam:

• Utilizar uma câmera IP com suporte à banda de 5 GHz acoplada ao robô ou encontrar uma forma de transmitir as imagens da câmera a uma taxa mais elevada com a Raspberry Pi por WiFi;

• Utilizar uma IMU mais precisa do que a do smartphone (controle da cabeça);

• Melhorar o a parte do código relacionada ao Kinect (controle dos braços e da caminhada) ou até mesmo substituí-lo por algum outro método como outras IMUs acopladas aos membros do teleoperador ou uma roupa de captura de movimentos;

• Modificar o código para melhorar o processamento paralelo garantindo que todas as partes do código sejam executadas em intervalos de tempo mais regulares, possivelmente substituindo o ROS por uma outra plataforma de programação com o uso de threads.

Com o que foi implementado, é possível utilizar o robô teleoperado para se locomover no ambiente, interagir com o ambiente e obter uma resposta do que está ocorrendo no ambiente ao redor do robô com a limitação da locomoção devido ao uso de um cabo Ethernet.

Como proposta futura para este trabalho de graduação seria a melhoria do que já foi implementado (código-fonte, problemas de conexão), estudar soluções alternetivas na implementação do código para uma execução mais regular de cada uma das suas partes, estudar a implementação de outras IMUs além da presente no smartphone, implementar um método para girar a mão do robô com o girar do pulso do teleoperador e encontrar uma aplicação que substitua o Trinus VR, eliminando a necessidade

de utilizar dois sistemas operacionais em 2 computadores ou uma máquina virtual.

[1] M. P. de Carvalho, "Controle de movimentação de humanóide em tempo real por teleoperação," 2014. [Online]. Available: http://www.ene.unb.br/antonio/resources/reports/mpcarvalho2014.pdf

[2] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," dec 2012. [Online]. Available: http://www.araa.asn.au/acra/acra2012/papers/pap126.pdf

[3] I. Rodriguez, A. Astigarraga, E. Jauregi, T. Ruiz, and E. Lazkano, "Humanizing nao robot teleoperation using ros," nov 2014.

[4] S. Goza, R. O. Ambrose, M. A. Diftler, and I. M. Spain, "Telepresence control of the nasa/darpa robonaut on a mobility platform," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 623–629.

[5] J. Hefferon, *Linear Algebra*, 2014.

[6] B. Kenwright, "A beginners guide to dual-quaternions." [Online]. Available: http://cs.gmu.edu/ jmlien/teaching/cs451/uploads/Main/dual-quaternion.pdf

[7] M. Morrison, "Inertial measurement unit," dec 1987, uS Patent 4,711,125. [Online]. Available: https://www.google.com/patents/US4711125

[8] "Gyrometer - colibrys - mems accelerometers," acessado em: 09 dez. 2016. [Online]. Available: http://www.colibrys.com/c/technology/memsgyro/

[9] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Tech. Rep., 2007. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf

[10] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1999.

[11] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[12] "Discover nao, the little humanoid robot from aldebaran," acesso em: 08 dez. 2016. [Online]. Available: https://www.ald.softbankrobotics.com/en/cool-robots/nao

[13] W. Zeng, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, 2012.

[14] "Myo gesture control armband," acesso em: 08 dez. 2016. [Online]. Available: https://www.myo.com/

[15] T. A. Kuiken, G. Li, B. A. Lock, R. D. Lipschutz, L. A. Miller, K. A. Stubblefield, and K. B. Englehart, "Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms," *JAMA*, 2009. [Online]. Available: http://jamanetwork.com/journals/jama/fullarticle/183371

[16] G.Senthilkumar, K.Gopalakrishnan, and V. S. Kumar, "Embedded image capturing system using raspberry pi system," apr 2014. [Online]. Available: http://ijettcs.org/Volume3Issue2/IJETTCS-2014-04-23-114.pdf

[17] J. Steuer, "Defining virtual reality: Dimensions determining telepresence," *Journal of Communication*, 1992. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1111/j.1460-2466.1992.tb00812.x/epdf

[18] "Google cardboard," acesso em: 08 dez. 2016. [Online]. Available: https://vr.google.com/cardboard/

[19] "Ros wiki: Documentation," acesso em: 08 dez. 2016. [Online]. Available: http://wiki.ros.org/

[20] "Naoqi key concepts," acesso em: 08 dez. 2016. [Online]. Available: http://doc.aldebaran.com/2-1/dev/naoqi/

[21] "Github - ccrisan/motioneyeos," acesso em: 08 dez. 2016. [Online]. Available: https://github.com/ccrisan/motioneyeos

[22] "Trinus vr," acesso em: 08 dez. 2016. [Online]. Available: http://trinusvr.com

[23] J. Postel, "User datagram protocol," aug 1980. [Online]. Available: https://tools.ietf.org/pdf/rfc768.pdf

[24] J. Sobotta, *Atlas de Anatomia Humana*, 21st ed., E. G. K. S.A., Ed., 2000, vol. Volume 1.

[25] "Locomotion control," acesso em: 08 dez. 2016. [Online]. Available: http://doc.aldebaran.com/1-14/naoqi/motion/control-walk.html

[26] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," *IEEE-RAS International Conference on Humanoid Robots*, 2006. [Online]. Available: https://hal.inria.fr/inria-00390462/document

[27] "The illusion of motion," acesso em: 09 dez. 2016. [Online]. Available: https://paulbakaus.com/tutorials/performance/the-illusion-of-motion/