Felipe Augusto Pires

**Avaliação de Métodos de Telereabilitação Robótica
utilizando comunicação TCP/IP e Unity**

Dissertação apresentada à Escola de Engenharia de São Carlos
da Universidade de São Paulo, como parte dos requisitos para
obtenção do título de Mestre em Engenharia Mecânica

Área de Concentração: Dinâmica de Máquinas e Sistemas
Orientador: Prof. Dr. Adriano A. G. Siqueira

ESTE EXEMPLAR TRATA-SE DA
VERSÃO CORRIGIDA.
A VERSÃO ORIGINAL ENCONTRA-
SE DISPONÍVEL JUNTO AO
DEPARTAMENTO DE
ENGENHARIA MECÂNICA DA
EESC-USP.

São Carlos
2014

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

P667a Pires, Felipe Augusto
Avaliação de Métodos de Telereabilitação Robótica
utilizando comunicação TCP/IP e Unity / Felipe Augusto
Pires; orientador Adriano Almeida Gonçalves Siqueira.
São Carlos, 2014.

Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia Mecânica e Área de Concentração em
Dinâmica de Máquinas e Sistemas -- Escola de Engenharia
de São Carlos da Universidade de São Paulo, 2014.

1. Reabilitação. 2. Robótica. 3. Telereabilitação.
4. Telecomunicação. 5. Unity. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **FELIPE AUGUSTO PIRES**.

Título da dissertação: "Avaliação de métodos de telereabilitação robótica utilizando comunicação TCP/IP e Unity".

Data da defesa: 05/02/2014

Comissão Julgadora:

Resultado:

Prof. Associado **Adriano Almeida Gonçalves Siqueira (orientador)**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **José Martins Junior**
(Escola de Engenharia de Piracicaba/EEP)

APROVADO

Prof. Dr. **Luiz Camolesi Júnior**
(Universidade Estadual de Campinas/UNICAMP)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Marcelo Areias Trindade**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

Sumário

Resumo	ix
Abstract	xi
Lista de Figuras	xiii
Lista de Tabelas	xix
1 Introdução	1
1.1 Objetivos	3
1.2 Disposição dos capítulos	3
2 Revisão Bibliográfica	5
2.1 Reabilitação Robótica	5
2.2 Telereabilitação Robótica	10
3 Metodologia e Desenvolvimento	15
3.1 Comunicação TCP/IP	15
3.1.1 JSON	17
3.1.2 Boost	18
3.2 Unity	19
3.2.1 Rendering	20
3.2.2 Scripting	20
3.2.3 Asset Tracking	21
3.2.4 <i>Physics</i>	21
3.2.5 Versão 4.X	21
3.3 Modelo Dinâmico na Unity	22
3.4 Teleoperação Robótica	23
3.4.1 Método 01: Controle Teleoperado Direto (PD Simétrico)	25
3.4.2 Método 02: Controle Teleoperado por Variáveis de Ondas	26

3.4.3	Método 03: Controle Teleoperado através de Ambiente Virtual	32
4	Ambiente de Jogos para Telereabilitação	35
4.1	Plataforma Virtual	35
4.2	Plataforma Háptica	43
4.3	Métodos de Teleoperação Robótica	46
4.3.1	Método 01: Controle Teleoperado Direto(PD Simétrico)	47
4.3.2	Método 02: Controle Teleoperado por Variáveis de Ondas	47
4.3.3	Método 03: Controle Teleoperado através de Ambiente Virtual	48
5	Resultados	51
5.1	Método 01: Controle Teleoperado Direto (PD Simétrico)	52
5.2	Método 02: Controle Teleoperado por Variáveis de Ondas	60
5.3	Método 03: Controle Teleoperado através de Ambiente Virtual	68
5.4	Tabela Comparativa	76
6	Conclusão	79
6.1	Pesquisas Futuras	80
	Referências Bibliográficas	81

Dedico a Deus, por me dar a força e a perseverança de conseguir chegar até esse momento da minha vida.

Dedico a todos os meus pares que tanto se esforçam para transformar ideias em ações e trazer benefícios a uma grande maioria.

Dedico por último e em especial a minha família, que sempre este presente apoiando da melhor maneira possível em minhas jornadas de superação.

Agradecimentos

Primeiramente, agradeço imensamente aos meus pais, que me deram muito amor e fizeram da vida deles um exemplo e uma ponte para que eu alcançasse todos os meus sonhos e metas. Sem eles eu nada seria.

Agradeço a minha amiga, companheira e eterno amor Fernanda, por todo apoio e dedicação nos momentos mais importantes dessa caminhada e por estar do meu lado por toda a vida.

Agradeço a Deus por ter tido saúde e força para cumprir essa jornada.

Agradeço ao meu orientador, Prof. Dr. Adriano A. G. Siqueira por todo companheirismo, apoio e dedicação durante essa jornada e principalmente pelo apoio nos momentos difíceis que passei durante a mesma. Sempre me fez crer que eu conseguiria.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, pela bolsa concedida durante os anos do curso.

Agradeço aos meus amigos Kleber de Oliveira Andrade, José Martins Junior, Ana Carolina Barbosa, Samuel Lourenço Nogueira, Carlos Eduardo Desuó, Breno Rocha Comin e a todos os colegas do grupo de reabilitação robótica, pelo apoio, amizade e enormes préstimos a este documento.

Agradeço também aos meus amigos do Senac: Carlos E. Pressutto, Fernando de Souza Pereira, Renata Tenorio Costa e Rodrigo de Magalhães L. Nunes pelo apoio em momentos complicados da jornada e em especial a minha amiga e coordenadora Isabela B. Carneiro por todo o suporte nos momentos cruciais dela.

Agradeço especialmente a todos os funcionários da EESC, pois direta ou indiretamente contribuíram de alguma forma para a assertividade desde trabalho.

“Ainda que eu falasse as línguas dos homens e dos anjos, e não tivesse amor, seria como o metal que soa ou como o sino que tine”

1 Coríntios 13:1

Resumo

PIRES, Felipe A. **Avaliação de Métodos de Telereabilitação Robótica utilizando comunicação TCP/IP e Unity.** 2014, 106 f. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2014. Este trabalho apresenta o desenvol-

vimento de um ambiente de estudos para Telereabilitação Robótica utilizando jogos cooperativos desenvolvidos com o ambiente de desenvolvimento Unity e comunicação via *sockets* baseados em TCP/IP. Para tanto, fez-se necessário a implementação de um programa de controle de comunicação, um jogo e um sistema robótico. A fim de estudar este ambiente, foram implementados três métodos de comunicação entre o sistema robótico e o jogo, de acordo com a literatura. Os métodos são: Controle Teleoperado Direto (PD Simétrico), Controle Teleoperado por Variáveis de Ondas e Controle Teleoperado através de Ambiente Virtual. Para realizar a avaliação destes métodos foram implementados quatro simulações de atraso de comunicação, onde os seguintes parâmetros foram avaliados: estabilidade das mensagens enviadas e recebidas, sincronismos dos motores do sistema robótico e seus parâmetros (torque e velocidade) e sincronismo do sistema robótico com o ambiente lúdico (jogo). As situações de atraso criadas foram de: 1 milissegundo, 300 milissegundos, 600 milissegundos e 900 milissegundos. Todos os métodos implementados se mostraram suscetíveis a altos índices de atraso (especialmente ao testarmos atrasos artificiais de 600 e 900 milissegundos). O método que apresentou menores falhas foi o de controle de variáveis de ondas.

Palavras-chave: Reabilitação, Robótica, Telereabilitação, Telecomunicação, Unity

Abstract

*PIRES, Felipe A. **Evaluation of Methods for Telerehabilitation Robotics using TCP/IP communication and Unity.**2014, 106 f. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2014. This dissertation presents the deve-*

lopment of an study environment for telerehabilitation robotics using cooperative games developed with Unity (Game Engine) and communication with sockets based on TCP / IP protocol. For this purpose, it was necessary to implement a control program of communication, one game and a robotic system. To study this environment, three methods of communication between the robotic system and the game according to the literature were implemented. These methods are: Direct Teleoperated Control (symmetrical PD) , Teleoperated Control for Wave Variables and Teleoperated Control through Virtual Environment . In order to evaluate these methods were impletamentados four simulations delay of communication , where the following parameters were evaluated : stability of messages sent and received , timings of the motors of the robotic system and its parameters (torque and speed) and timing of the robotic system with playful environment (game). The delay simulations were created: 1 millisecond, 300 milliseconds, 600 milliseconds and 900 milliseconds. All implemented methods proved susceptible to high rates of delay (especially when we test artificial delays of 600 and 900 milliseconds). The method presented minor flaws was the Teleoperated Control for Wave Variables

Keywords: rehabilitation, robotics, telerehabilitation, telecommunication, Unity

Lista de Figuras

FIGURA 2.1	Robôs interativos desenvolvidos no Newman laboratory for Biomechanics and Human Rehabilitation. Fonte: (HOGAN (1985))	7
FIGURA 2.2	Lokomat. Fonte: (JEZERNIK ET AL. (2003))	7
FIGURA 2.3	Anklebot. Fonte: (ROY ET AL. (2009))	9
FIGURA 2.4	Testes com o Anklebot e imagens do jogo computacional. Fonte: (FORRESTER ET AL. (2011))	10
FIGURA 2.5	Estrutura de telereabilitação robótica unilateral, adaptada de Carignan e Krebs (2006).	11
FIGURA 2.6	Estrutura de telereabilitação robótica bilateral, adaptada de Carignan e Olsson (2004); Carignan e Krebs (2006).	12
FIGURA 3.1	Utilização de <i>sockets</i> em comunicação via internet.	16
FIGURA 3.2	Configurações de redes de comunicação via internet.. Fonte: (HOGAN (1985))	16
FIGURA 3.3	Modelo de construção do JSON - Adaptado de http://www.json.org/json-pt.html . Acessado em 20 de julho de 2013	17
FIGURA 3.4	Modelo Virtual no ambiente de desenvolvimento Unity.	22
FIGURA 3.5	Parâmetros de configuração do modelo virtual.	23
FIGURA 3.6	Sistema robótico teleoperado.	24
FIGURA 3.7	Controle PD simétrico.	25
FIGURA 3.8	Exemplo de aplicação da transformada de onda.	27

FIGURA 3.9	Descrição do sistema no domínio das variáveis de onda.	27
FIGURA 3.10	Sistema teleoperado básico baseado em variáveis de onda.	29
FIGURA 3.11	Sistema teleoperado com controlador PD no robô escravo e aplicação direta de força no mestre.	30
FIGURA 3.12	Sistema teleoperado com casamento de impedâncias.	31
FIGURA 3.13	Estrutura de comunicação do sistema teleoperado baseado em modelo virtual.	33
FIGURA 4.1	Objeto virtual de simulação da prancha.	36
FIGURA 4.2	Plataforma Virtual desenvolvida na Unity	36
FIGURA 4.3	Uso do modelo dinâmico da Unity.	37
FIGURA 4.4	Objetos virtuais de simulação das manoplas.	40
FIGURA 4.5	Plataforma Háptica	43
FIGURA 4.6	Detalhes da EPOS da Plataforma Háptica	44
FIGURA 4.7	Sistema de comunicação da Plataforma Háptica com a Plataforma Virtual	44
FIGURA 4.8	Diagrama do Método 01 - Sem controle de Variáveis de Onda e sem Modelo Dinâmico da Unity	47
FIGURA 4.9	Diagrama do Método 02 - Com controle de Variáveis de Onda e sem Modelo Dinâmico da Unity	48
FIGURA 4.10	Diagrama do Método 03 - Sem controle de Variáveis de Onda e com Modelo Dinâmico da Unity	49
FIGURA 5.1	Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 01.	52
FIGURA 5.2	Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 01.	53
FIGURA 5.3	Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 01.	53

FIGURA 5.4	Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 01.	54
FIGURA 5.5	Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 300.	54
FIGURA 5.6	Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 300.	55
FIGURA 5.7	Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 300.	55
FIGURA 5.8	Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 300.	56
FIGURA 5.9	Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 600.	56
FIGURA 5.10	Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 600.	57
FIGURA 5.11	Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 600.	57
FIGURA 5.12	Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 600.	58
FIGURA 5.13	Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 900.	58
FIGURA 5.14	Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 900.	59
FIGURA 5.15	Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 900.	59
FIGURA 5.16	Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 900.	60
FIGURA 5.17	Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 01.	60

FIGURA 5.18 Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 01.	61
FIGURA 5.19 Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 01.	61
FIGURA 5.20 Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 01.	62
FIGURA 5.21 Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 300.	62
FIGURA 5.22 Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 300.	63
FIGURA 5.23 Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 300.	63
FIGURA 5.24 Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 300.	64
FIGURA 5.25 Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 600.	64
FIGURA 5.26 Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 600.	65
FIGURA 5.27 Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 600.	65
FIGURA 5.28 Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 600.	66
FIGURA 5.29 Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 900.	66
FIGURA 5.30 Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 900.	67
FIGURA 5.31 Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 900.	67

FIGURA 5.32 Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 900.	68
FIGURA 5.33 Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 01.	68
FIGURA 5.34 Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 01.	69
FIGURA 5.35 Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 01.	69
FIGURA 5.36 Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 01.	70
FIGURA 5.37 Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 300.	70
FIGURA 5.38 Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 300.	71
FIGURA 5.39 Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 300.	71
FIGURA 5.40 Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 300.	72
FIGURA 5.41 Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 600.	72
FIGURA 5.42 Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 600.	73
FIGURA 5.43 Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 600.	73
FIGURA 5.44 Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 600.	74
FIGURA 5.45 Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 900.	74

FIGURA 5.46 Posição dos Motores em relação ao tempo de resposta - Método 03 com
atraso igual a 900. 75

FIGURA 5.47 Relação de Torque x Tempo entre os dois motores - Método 03 com
atraso igual a 900. 75

FIGURA 5.48 Velocidade dos Motores relacionada ao tempo - Método 03 com atraso
igual a 900. 76

Lista de Tabelas

TABELA 5.1	Tabela comparativa dos métodos de teleoperação robótica.	78
------------	--	----

Capítulo 1

Introdução

Segundo o último censo brasileiro divulgado recentemente pelo Instituto Brasileiro de Geografia e Estatística (IBGE) (2010), houve um crescimento expressivo da parcela relativa da população com 65 anos ou mais, passando de 4,8% em 1991 para 5,9% em 2000 e 7,4% em 2010. Ou seja, o número de idosos hoje no Brasil chega a quase 18 milhões - 12% da população total. Levando em consideração que a incidência de Acidente Vascular Encefálico (AVE) aumenta com a idade, estes dados trazem um alerta preocupante.

No Brasil, em torno de 40% das mortes são causadas por doença cardiovascular (BENSEÑOR E LOTUFO (2008)), sendo que a mortalidade por AVE (10%) predomina em relação à mortalidade por infarto do miocárdio (9%). Os números atingem em torno de 100 mil vítimas por ano. Segundo Abramczuk e Villela (2009), o Sistema Único de Saúde (SUS) registrou, no ano de 2008 cerca de 200 mil internações por AVE, com 33 mil casos evoluindo para óbito.

O AVE além do óbito, pode levar a sequelas graves que atingem em torno de 50% dos sobreviventes. Embora a melhoria dos medicamentos e suprimentos desenvolvidos para a proteção e recuperação neural pós-AVE (bloqueadores de receptores, anti-oxidantes e trombolíticos) possa eventualmente aumentar a taxa de sobrevivência e diminuir a severidade do AVE, o número de pessoas que necessitarão de reabilitação aumentará significativamente com o aumento da população de risco.

Entre as sequelas decorrentes de um AVE, os danos às áreas neurais que controlam o movimento podem gerar um conjunto de deficiências motoras e sensoriais centrais e periféricas. Estas incluem a diminuição da amplitude de movimento dos membros contralaterais ao lado da lesão, fraqueza (GANDEVIA (1993)), reflexos hiperativos (THILMANN ET AL. (1993)) e desco-

ordenação de movimentos, que podem se manifestar como uma incapacidade de ativar independentemente os músculos necessários para realizar as atividades motoras funcionais (DEWALD ET AL. (1995)).

De uma forma geral, indivíduos pós-AVE conseguem recuperar certos movimentos de maneira espontânea. Entretanto, um programa de terapias física e ocupacional é sempre necessário para que a recuperação ocorra de maneira rápida e plena. As terapias pós-AVE geralmente requerem um trabalho intensivo por parte do paciente e do fisioterapeuta, que na maioria dos casos atende apenas um paciente por vez, tendo que trabalhar durante vários períodos do dia (TROMBLY (2002)).

Novos tratamentos e soluções tecnológicas foram desenvolvidas para lidar com os desafios da reabilitação pós-AVE: terapia por contensão induzida, que tenta aperfeiçoar a função motora do membro parético combinando um programa de treinamento intensivo com a contensão do membro superior não afetado (TAUB E ELBERT (2002); VAN DER LEE ET AL. (1999)); aumento da intensidade e duração da terapia incluindo manipulação externa (DAM ET AL. (1993)); treinamento do movimento bilateral (WHITEALL ET AL. (2000)); e a reabilitação robótica (FASOLI ET AL. (2003); KREBS ET AL. (2008, 2003); RIENER ET AL. (2005)). O mecanismo comum a todas as soluções propostas nestas pesquisas se baseia no movimento repetitivo e estruturado, que presumidamente promove a reorganização cortical devido à característica de neuroplasticidade apresentada pelos seres humanos (NUDO E FRIEL (1999); STAINES ET AL. (2001)).

Iniciada nos anos 90, a reabilitação assistida por computadores tem avançado muito nos últimos anos devido a algumas mudanças de paradigmas. De modo geral a tecnologia por trás desse tipo de reabilitação passou de meramente assistiva para ser utilizada como procedimentos que visam o aumento de produtividade e eficácia dos métodos terapêuticos e do próprio terapeuta. Este aumento destaca-se também como uma solução viável para redução de custos em tratamentos, como o AVE por exemplo.

A reabilitação robótica quando somada a telemedicina (uso de telecomunicação em conjunto com a medicina), se apresenta como ferramenta para proporcionar aos pesquisadores a possibilidade de desenvolver novas terapias de reabilitação (CARIGNAN E KREBS (2006)). A telereabilitação robótica, considerada aqui como uma reabilitação remota, pode ser realizada sob duas possibilidades: unilateral e bilateral.

Unilateral se dá quando apenas o paciente utiliza o robô e o terapeuta acompanha o seu

desempenho por meio das informações transmitidas via internet, possibilitando ao terapeuta remotamente alterar as configurações do robô ou do jogo computacional que o paciente está trabalhando no momento.

Quando tanto o paciente como o terapeuta (ou outro paciente) utilizam os robôs e comunicam suas informações entre si, tem-se um sistema bilateral, onde o terapeuta pode sentir e alterar o comportamento do paciente, por meio das ações diretas dos movimentos dos robôs.

Ao unificarmos todas estas tecnologias em um único ambiente, temos uma proposta de melhoria e aumento de produtividade extremamente relevantes para o tratamento de indivíduos pós-AVE.

1.1 Objetivos

O presente trabalho tem como principal objetivo desenvolver um ambiente de comunicação de jogos para telereabilitação robótica baseado na Unity (ambiente de desenvolvimento para jogos) e comunicação TCP/IP. Afim de garantir a estabilidade dessa comunicação, este trabalho também utiliza variáveis de ondas como algoritmo primário de comunicação entre a plataforma háptica e ambiente lúdico (jogo), e através do estudo de 3 casos de comunicação, consegue-se comprovar a sua eficácia.

1.2 Disposição dos capítulos

Este documento está organizado como apresentado a seguir. O **Capítulo 2** descreve a revisão de estudos das áreas Reabilitação Robótica e Telereabilitação Robótica e apresenta modelos propostos na literatura. No **Capítulo 3** são apresentadas as tecnologias utilizadas para o desenvolvimento do ambiente proposto, bem como os métodos de comunicação aplicados para a telereabilitação. No **Capítulo 4** descreve-se o ambiente de desenvolvimento proposto para jogos baseados em telereabilitação, que consiste em uma plataforma háptica, e o seu modelo virtual, todo baseado no modelo dinâmico da Unity, a integração desse modelo com o sistema robótico via TCP/IP e os modelos de teleoperação robótica. Os dados obtidos dos testes da plataforma háptica em relação aos métodos de comunicação propostos estão elencados no **Capítulo 5**. As conclusões e sugestões para futuros trabalhos são apresentadas no **Capítulo 6**.

Capítulo 2

Revisão Bibliográfica

Neste capítulo desenvolveu-se um estudo do estado da arte sobre os principais temas que permeiam este documento, Reabilitação Robótica e Telereabilitação Robótica, com a finalidade de elencar suas características e particularidades e promover um entendimento do quão importante é o estudo e o desenvolvimento de aplicabilidades dos mesmos.

2.1 Reabilitação Robótica

A aplicação de robôs em reabilitação data do início dos anos 90, entretanto, o aumento significativo de trabalhos nesta área nos últimos anos deve-se à mudança de paradigma observada: de uma tecnologia assistiva para pessoas com deficiência para procedimentos baseados na terapia robótica, que utiliza a tecnologia para suportar e aumentar a produtividade e eficiência dos terapeutas, visando a recuperação dos indivíduos. Este aumento de produtividade e eficiência destaca-se como uma solução viável para a reabilitação de pacientes pós-AVE, uma vez que há um aumento desses casos e, como consequência, maiores custos de tratamento. A Associação Americana do Coração (*American Heart Association*, (ASSOCIATION (2011))) estimou em US\$ 40 bilhões os gastos diretos, que incluem custos com médicos e outros profissionais, medicações, serviços hospitalares e cuidados na residência, e indiretos gerados pela perda de produtividade em função de morte ou invalidez prematura, nos EUA em 2007. Somente no atendimento inicial, um paciente com AVE custa para o Sistema Único de Saúde (SUS), em média, R\$ 6 mil, mas esse valor pode variar de acordo com a gravidade de cada caso: de R\$ 640,00 para um paciente que se recupera em um período de internação curto de 3 a 5 dias a R\$ 32 mil para pacientes com sequelas graves com período de internação de mais de um mês (ABRAMCZUK E VILLELA

(2009)). Com essas demandas no sistema de saúde, a redução de custos é uma demanda urgente, não só pelo tempo de internação dos pacientes, mas também pelo encaminhamento de pacientes para serviços especializados, que nem sempre oferecem um atendimento adequado para a reabilitação.

Muitos centros de pesquisa ao redor do mundo buscam adaptar ou reconfigurar robôs industriais para o uso em reabilitação (HARWIN ET AL. (2001); LUM ET AL. (1995)). Esta estratégia, embora pareça ser razoável, não é apropriada, pois a literatura mostra que impedâncias mecânicas baixas, comparáveis às de seres humanos, não podem ser alcançadas com estes sistemas. A impedância mecânica é a razão de um potencial (por exemplo, força) a um fluxo (por exemplo, velocidade) em que os argumentos das partes real (ou imaginários) de ambos aumentam linearmente com o tempo. A impedância é recíproca à mobilidade, ou seja, quanto maior velocidade de movimentação, maior deve ser a força de impedância aplicada pelo sistema e o inverso também é válido.

Robôs comerciais utilizam, em baixo nível, o controle de posição, que não produz resultados satisfatórios quando forças de contato estão presentes. Controle de força por realimentação pode melhorar a resposta do robô, mas não é suficiente para produzir baixa impedância mecânica, necessária para mover suavemente e rapidamente o ponto de contato em complacência com as ações do paciente (LAWRENCE (1988)).

Por outro lado, o controle de impedância, proposto inicialmente por Neville Hogan, em 1985 (HOGAN (1985)), atua diretamente na especificação da interface de contato entre o robô e o ambiente e soluciona este problema. O controle de impedância também modula a forma como o sistema robótico reage a perturbações geradas pelo paciente e garante um comportamento complacente. Uma baixa impedância significa que o robô se deslocará à medida que o paciente exercer força sobre ele, característica denominada *backdriveability*. Portanto, um sistema robótico com controle de impedância pode ser programado para permitir que um paciente se recuperando de AVE realize movimento no todo ou em parte, mesmo quando a tentativa for fraca ou descoordenada. Neste caso, o robô auxilia o paciente a completar a tarefa. Os pioneiros na utilização do controle de impedância na reabilitação robótica foram os pesquisadores do Newman Laboratory for Biomechanics and Human Rehabilitation, do Massachusetts Institute of Technology (MIT), do qual fazem parte os Profs. Neville Hogan e Hermano I. Krebs. Inicialmente, foi produzido o robô MIT-MANUS (InMotion Arm Robot), Figura 2.1(a), projetado para reabilitação dos membros superiores, em especial o ombro e cotovelo. Também foram de-

envolvidos robôs interativos para o punho, comercializados como InMotion Wrist Robot (Figura 2.1(b)) e tornozelo.

(a) MIT-MANUS (InMotion Arm Robot)

(b) InMotion Wrist Robot

Figura 2.1: Robôs interativos desenvolvidos no Newman laboratory for Biomechanics and Human Rehabilitation. Fonte: (HOGAN (1985))

Considerando a reabilitação do caminhar, em Jezernik et al. (2003), uma órtese robótica, denominada Lokomat, é utilizada na reabilitação de pacientes com lesão medular ou que sofreram AVE, Figura 2.2.

Figura 2.2: Lokomat. Fonte: (JEZERNIK ET AL. (2003))

O dispositivo é instalado sobre uma esteira rolante e o paciente se locomove utilizando um compensador de peso. O treinamento regular de pacientes é realizado impondo um padrão de marcha fixo através de um controle de posição das articulações da órtese robótica. Porém, é importante garantir que o paciente esteja efetivamente andando, e não apenas tendo a sua perna movida passivamente pelo aparelho de locomoção. Esta condição levou ao desenvolvimento de três algoritmos de adaptação do padrão de marcha (JEZERNIK ET AL. (2004); RIENER ET AL.

(2005)). Esses algoritmos possibilitam ao paciente alterar o padrão de marcha conforme o seu grau de locomoção voluntária. Entretanto, até o momento não foram relatados estudos que mostrem um desempenho superior desta abordagem baseada em padrões cinemáticos em comparação à terapia convencional (AHN E HOGAN (2010)). Mesmo novos algoritmos que utilizam o controle de impedância e que foram recentemente incluídos no Lokomat necessitam ter sua eficácia comprovada, principalmente considerando o elevado custo de tais equipamentos.

Da mesma forma não foram relatados estudos que mostrem um desempenho superior da terapia robótica em comparação à terapia convencional (HIDLER ET AL. (2009); HORNBY ET AL. (2008)). Uma possível explicação para estes fatos é que a maioria das terapias propostas assume que a recuperação dos padrões cinemáticos do movimento das pernas é suficiente para a recuperação das funções locomotoras. Neste caso, os robôs impõem padrões cinemáticos por meio do seguimento de uma trajetória pré-definida (AHN E HOGAN (2010)). Entretanto, na reprodução de tais padrões, não são considerados os efeitos da gravidade e do sistema neuromuscular periférico, interferindo na dinâmica natural do caminhar.

Buscando uma estratégia diferente, em Roy et al. (2009) foi desenvolvido um dispositivo robótico para reabilitação do tornozelo, denominado Anklebot, que segue a mesma linha dos robôs para membros superiores desenvolvidos pelo mesmo grupo, Figura 2.3. Ele possui baixo atrito e utiliza o controle de impedância para garantir que o sistema seja *backdrivable*. O sistema possibilita a movimentação nos três graus de liberdade do pé com relação à perna, sendo que dois deles são atuados por um par de atuadores lineares montados em paralelo. Os movimentos atuados são: dorsiflexão e flexão plantar no plano sagital (junta DP), obtidos quando os dois atuadores se movimentam no mesmo sentido; e inversão e eversão no plano frontal (junta IE), quando os dois atuadores se movimentam em sentidos opostos. Utilizar menos atuadores que os graus de liberdade das juntas do pé permite que o dispositivo seja instalado sem a necessidade de se ter um alinhamento preciso entre as juntas do robô e as juntas do paciente.

O Anklebot pode aplicar até 23 Nm em dorsiflexão/flexão plantar e 15 Nm em inversão/eversão. Embora estes valores não sejam suficientes para suportar o peso do paciente, o objetivo do projeto do dispositivo era fornecer um torque complementar ao membro parético para amenizar os efeitos do pé caído (*drop foot*, em inglês). Esta deficiência, causada pela fraqueza dos músculos responsáveis pela dorsiflexão e presente na maioria dos pacientes hemiparéticos pós-AVE, apresenta duas complicações no caminhar: após o contato do calcanhar com o chão, a ponta do pé cai rapidamente e durante a fase de balanço a ponta do pé toca o chão. Além de

Figura 2.3: Anklebot. Fonte: (ROY ET AL. (2009))

controlar a posição da junta do tornozelo nestes instantes, realizando a dorsiflexão, o Anklebot permite auxiliar o paciente no final da fase de suporte, aplicando torque durante a flexão plantar necessária para dar o impulso do caminhar e retirar o pé do chão. O Anklebot também pode ser utilizado por crianças de 6 a 12 anos com paralisia cerebral, sendo que uma versão para crianças de 2 a 6 anos está sendo desenvolvida e será construída em breve.

Resultados preliminares da utilização do Anklebot para reabilitação foram apresentados em Forrester et al. (2011). Foi realizado um estudo piloto com oito pacientes sobreviventes de AVE, submetidos a três sessões por semana, de 1 (uma) hora cada, nas quais os pacientes exercitaram sentados os movimentos de dorsiflexão e flexão plantar. Os movimentos eram realizados em função da solicitação imposta por um jogo computacional (*video game*), no qual um cursor deve ser posicionado alinhado com o orifício de uma parede que se move para a esquerda, Figura 2.4. Foram avaliados a amplitude de movimento do membro parético, a força realizada, o controle motor e o caminhar em solo antes e depois de seis semanas de reabilitação. Os dados mostram que houve uma melhora no caminhar, com aumento da velocidade e no controle motor. Salienta-se que os procedimentos de reabilitação foram realizados com os pacientes sentados.

Em Ahn e Hogan (2010), características dinâmicas naturais do caminhar foram exploradas buscando gerar uma estratégia para o treinamento assistido por robôs. Evidências indicam a existência de um gerador de padrões central (CPG, do inglês *Central Pattern Generator*) que fornece sinais periódicos que regulam o caminhar humano. Além disso, entradas relacionadas à carga e ao tato provenientes do pé e do tornozelo influenciam na resposta do CPG (RYBAK ET AL. (2006)). Utilizando-se destas premissas, os autores aplicaram pulsos periódicos de torque no plano sagital do dispositivo robótico Anklebot e realizaram testes com pessoas sem

Figura 2.4: Testes com o Anklebot e imagens do jogo computacional. Fonte: (FORRESTER ET AL. (2011))

patologias, sem que estes soubessem da aplicação do torque. Verificou-se que, dentro de certos limites, o caminhar dos usuários apresentou o mesmo período do sinal de entrada. Além disso, o usuário realizou a flexão plantar no final da fase de suporte no instante da aplicação do torque pelo Anklebot. Ou seja, o torque aplicado auxiliou o usuário a realizar o caminhar, uma característica importante para pacientes que não podem produzir a propulsão necessária.

2.2 Telereabilitação Robótica

A Telemedicina tem sido definida como o uso de telecomunicações para troca de informações entre pacientes, médicos e outros profissionais da saúde, além de viabilizar serviços diretos para os pacientes. A Telereabilitação relaciona-se com a telemedicina, mas está além, pois demanda mais recursos do que somente transmissão de voz e imagem. Neste contexto, a reabilitação robótica, descrita na seção anterior, proporciona aos pesquisadores a possibilidade de desenvolver novas terapias de reabilitação (CARIGNAN E KREBS (2006)).

A Telereabilitação Robótica, considerada aqui como uma reabilitação remota, pode ser realizada de forma unilateral, ou seja, apenas o paciente utiliza o robô e o terapeuta acompanha o seu desempenho por meio das informações transmitidas via Internet. Nesta configuração, o terapeuta pode, remotamente, alterar as configurações do robô ou do jogo computacional que o paciente está trabalhando no momento, Figura 2.5. Quando tanto o paciente como o terapeuta (ou outro paciente) utilizam os robôs e comunicam suas informações entre si, temos um sistema bilateral, Figura 2.6. Neste caso, o terapeuta pode sentir e alterar o comportamento do paciente, por meio das ações diretas dos movimentos dos robôs. A proposta deste trabalho insere-se na

telereabilitação robótica bilateral.

Figura 2.5: Estrutura de telereabilitação robótica unilateral, adaptada de Carignan e Krebs (2006).

Em Carignan e Krebs (2006), foram descritas duas variações da telereabilitação bilateral de acordo com o nível de interação entre os usuários dos robôs: a telereabilitação interativa e a cooperativa. Na telereabilitação interativa, os dois usuários interagem entre si por meio do ambiente gráfico do jogo, mas não há transmissão de força direta entre os dois robôs. Cada indivíduo movimentava de forma independente o seu robô, cuja posição está associada a um objeto do ambiente gráfico do jogo, por exemplo, a prancha de madeira que equilibra a bola.

Na telereabilitação cooperativa, paciente e terapeuta (ou dois pacientes) interagem tanto visualmente como “fisicamente”, pela percepção, embora de forma indireta através dos robôs, da força realizada pelo outro jogador. Para viabilizar a implementação desta forma de terapia bilateral, fez-se necessário estudar conceitos relacionados a teleoperação de robôs.

A teleoperação de robôs é um campo de pesquisa mais amplo, pois pode ser aplicada tanto para movimentação de objetos em ambientes hostis de forma remota, como por exemplo nas explorações espaciais ou em usinas nucleares, como também para a realização de cirurgias endoscópicas teleoperadas (como por exemplo o *da Vinci Surgical System*, www.davincisurgery.com). Na teleoperação tradicional, um operador humano conduz uma tarefa através da aplicação de força em um robô mestre. A posição resultante do robô mestre é transferida e replicada no robô escravo. Quando o robô escravo entra em contato com o ambiente remoto, a força de contato deve ser refletida diretamente para o operador, melhorando o desempenho da tarefa. Se a operação de teleoperação é realizada com uma comunicação sujeita a atrasos excessivos, o sistema teleoperado pode ficar instável.

Figura 2.6: Estrutura de telereabilitação robótica bilateral, adaptada de Carignan e Olsson (2004); Carignan e Krebs (2006).

Os primeiros estudos de teleoperação que levam em conta os efeitos do atraso na estabilidade do sistema remontam da década de 80. Em Anderson e Spong (1989), os autores utilizam conceitos de redes de telecomunicações para garantir a estabilidade do sistema teleoperado mesmo na presença de atraso. Uma análise da geração e transmissão de potência entre os robôs é realizada e critérios baseados em passividade são utilizados para garantir a estabilidade. Estendendo os resultados de Anderson e Spong (1989), em Niemeyer (1996) é introduzido o conceito de variáveis de onda, que fornece um *framework* natural para a análise e implementação de sistemas teleoperados com atrasos. A utilização de variáveis de onda - ou a transformada de ondas - garante que a comunicação entre os robôs seja passiva, fazendo com que o sistema seja estável. Mais detalhes podem ser obtidos nas referências (NIEMEYER E SLOTTINE (1997A,B, 1998)).

Neste trabalho, duas estratégias de controle de sistema teleoperado descritas em Niemeyer (1996) foram implementadas no sistema de telereabilitação robótica desenvolvido para dois Anklebots. Na primeira estratégia, denominada controle PD simétrico, as posições e velocidades de um robô são utilizadas como posições e velocidades desejadas para o outro robô, e vice-versa. Esta configuração mostra-se instável na presença de atrasos elevados. Utilizando as variáveis de onda para a transmissão da informação entre os robôs, a segunda estratégia realiza o casamento

de impedâncias dos dois sistemas robóticos, de tal forma que o sistema teleoperado é estável mesmo com atrasos excessivos.

Também foi estudada e implementada, embora de forma inicial, a estratégia de telereabilitação robótica proposta em Carignan e Olsson (2004) e representada na Figura 2.6. Nesta configuração, um modelo virtual do objeto sendo transportado pelos usuários é simulado em um dos computadores. As forças geradas pelos usuários e medidas pela interação com os robôs são aplicadas no modelo virtual, gerando uma nova posição para este. Esta nova configuração do objeto é transmitida de volta aos robôs, fazendo com que os pacientes sintam a movimentação do objeto. As transmissões entre os robôs e o modelo virtual também são realizadas por meio de variáveis de onda para garantir a estabilidade do sistema completo. Mais detalhes podem ser obtidos nas referências (OLSSON ET AL. (2004); TANG ET AL. (2005)).

Capítulo 3

Metodologia e Desenvolvimento

No capítulo em questão, descreve-se todos os métodos utilizados para o desenvolvimento deste trabalho.

3.1 Comunicação TCP/IP

A comunicação via Internet foi realizada por meio de *sockets* utilizando-se o protocolo TCP/IP. Após um estudo sobre protocolos de internet, verificou-se que dois protocolos de transporte são mais comumente utilizados sobre o IP (*Internet Protocol*): o TCP (*Transmission Control Protocol*) e o UDP (*User Datagram Protocol*). Embora o protocolo UDP seja mais rápido, suas funcionalidades não garantem confiabilidade na entrega dos dados transmitidos. Por outro lado, o protocolo TCP é confiável, pois possui controle de fluxo e congestionamento, checagem de dados, entre outras funcionalidades. Por esta razão, optou-se pela utilização deste protocolo.

O termo *socket* se refere a um canal de comunicação entre os protocolos de Internet e as aplicações, sendo identificado pelo protocolo utilizado e pelo endereço local ou remoto. Uma dada aplicação pode acessar diferentes *sockets*, sendo que um único também pode ser acessado por várias aplicações, Figura 3.1. A comunicação por meios de *sockets* se dá considerando a estrutura Cliente/Servidor. Neste caso, tanto o cliente como o servidor criam os seus respectivos *sockets*. No computador servidor, o *socket* é configurado para verificar continuamente se algum cliente está solicitando uma conexão. Em caso positivo, o servidor aceita a conexão e começa a comunicar com o cliente. A comunicação só é interrompida quando a conexão é finalizada em

um dos dois extremos.

Figura 3.1: Utilização de *sockets* em comunicação via internet.

Existem diferentes configurações de comunicação entre servidores e clientes, como por exemplo, a configuração centralizada, na qual apenas um computador é o servidor e os demais são clientes, Figura 3.2(a) ou a configuração *peer-to-peer* na qual todos os computadores podem assumir tanto o papel de servidor como o de cliente, Figura 3.2(b). Para a aplicação específica de apenas dois sistemas robóticos, o caso mais simples se aplica, quando um computador é o servidor e o outro é o cliente.

(a) Configuração de rede centralizada

(b) Configuração de rede *peer-to-peer*

Figura 3.2: Configurações de redes de comunicação via internet.. Fonte: (HOGAN (1985))

3.1.1 JSON

O JSON é um acrônimo para *JavaScript Object Notation* e é um formato leve de texto para a troca de informações entre aplicativos diversos. Está baseado em um subconjunto da linguagem de programação JavaScript (Standard ECMA-262, 3a Edição, Dezembro 1999). Por ser um formato de texto, o JSON é completamente independente de linguagem, pois utiliza das convenções que são familiares à linguagem C e a suas similares.¹ O formato JSON foi originalmente criado por Douglas Crockford e é descrito no RFC 4627.

JSON é construído com base em uma coleção de pares chave/valor, que definem as propriedades e seus valores da seguinte forma: inicia-se com { (chave aberta) e é finalizado com } (chave fechada) . Cada chave é seguida de : (dois pontos) e os pares são separados por , (vírgula).

Figura 3.3: Modelo de construção do JSON - Adaptado de <http://www.json.org/json-pt.html>. Acessado em 20 de julho de 2013

O JSON pode representar quatro tipos primários de dados (*strings*, números, booleanos e nulos) e dois tipos estruturados (objetos e vetores). Um objeto é uma coleção não ordenada de zero ou mais pares nome/valor, na qual o nome é uma *string* e o valor é uma *string*, número, booleano, nulo, objeto ou vetor. Um vetor é uma sequência ordenada de zero ou mais valores.

As vantagens do JSON sobre outros formatos de troca de dados (XML por exemplo) são: a diminuição do tamanho da resposta, a facilidade de entendimento para o programador (já que o código é similar a linguagem), e principalmente, no caso de Javascript, o *parser* facilitado porque é baseado em sua especificação de objeto. Usado como configuração em aplicações no lado servidor, pode-se construir uma arquitetura com base em interfaces para validar os dados, assim como validaria usando *schemas* no XML. O formato pode ser usado para trocar informações por aplicativos usando as seguintes linguagens: ActionScript, C, C, ColdFusion, Common Lisp, E, Erlang, Java, JavaScript, Lua, Objective CAML, Perl, PHP, Python, Rebol, Ruby, e Scheme.

¹(Em: <http://www.json.org/json-pt.html>. Acesso em: 20 julho 2013.)

3.1.2 Boost

Boost é uma coleção de bibliotecas que estendem a funcionalidade da linguagem de programação C++. Esta coleção possui uma licença especial, desenvolvida para ser utilizada em qualquer projeto. Várias bibliotecas presentes na Boost já foram aceitas no *Technical Report 1* da próxima especificação da biblioteca padrão do C++1 (cujo nome é informalmente conhecido como C++0x2), sendo que vários dos fundadores da Boost estão no comitê de padronização da linguagem.

Devido à ampla revisão e controle de qualidade, as bibliotecas incluídas possuem grande qualidade, sendo destinadas a uma vasta gama de usuários do C++, em diversos domínios. As possibilidades passam desde bibliotecas gerais como *smart_ptr*³ e abstrações do sistema operacional como o *filesystem*⁴ até bibliotecas para usuários avançados como a MPL⁵.

Para garantir eficiência e flexibilidade, o Boost faz uso extensivo de modelos, tornando-se uma fonte de pesquisa para o desenvolvimento de programação genérica e meta programação em C++. Para utilizá-la, o desenvolvedor deve estar familiarizado com o estilo C++, isto é, a tendência em utilizar o C++ como um linguagem própria ao invés de uma extensão do C.²

O Boost fornece bibliotecas nas seguintes áreas:³

- Algoritmos;
- Auxiliares para compiladores que não seguem o padrão;
- Programação concorrente (utilização de *threads*);
- *Containers*;
- Codificação correta e teste;
- Estrutura de dados;
- Funções-objeto e programação de alta-ordem;
- Programação genérica;
- Grafos;
- Entrada/Saída de dados;

²(Em: <http://www.boost.org/>. Acesso em: 16 julho 2013.)

³Informações fornecidas pelo site do fabricante

- Suporte inter-linguagem;
- Iteradores;
- Matemática e sistemas numéricos;
- Utilização e gerenciamento de memória;
- Análise sintática (através da plataforma Spirit);
- Meta-programação de pré-processadores;
- Meta-programação de modelos;
- Ponteiros inteligentes;
- Processamento de cadeias de caracteres;

3.2 Unity

Unity (também conhecido como Unity3D) é um ambiente de desenvolvimento de jogos multi-plataforma desenvolvida pela Unity Technologies. Este ambiente conta com um conjunto de ferramentas que permite criar conteúdos interativos em três dimensões, conteúdos estes que vão desde jogos a conteúdos educativos ou simulações em áreas de ciência. Podemos dividir estas ferramentas em: um motor de renderização, uma ferramenta completa com *workflows*, publicação em multiplataformas, simulações físicas e uma comunidade para compartilhamento de conhecimentos. Na área de jogos, este ambiente é capaz de criar aplicações para os mais diversos mercados, tais como: o mercado Web, plataformas *Desktop*, consoles e dispositivos móveis. A versão mais atual da Unity é a 4.3, sendo utilizada por mais de um milhão de desenvolvedores. Essa versão suporta desenvolvimento para as seguintes plataformas: iOS, Android, Windows, Blackberry 10, OS X, Linux, navegadores Web, Flash, PlayStation 3, Xbox 360, Windows Phone 8, e Nintendo Wii U.

Uma das grandes características que rendeu a grande aceitação da Unity é estar apta a trabalhar com os principais sistemas operacionais do mercado (Windows, Linux e Mac OS). Além disso, ela possui duas versões de trabalho: uma gratuita que pode ser utilizada tanto para fins educacionais quanto comerciais - desde que a empresa possua um lucro menor que U\$100.000,00 em seu último ano fiscal - e a versão paga, denominada Unity Pro, que contém

recursos específicos e aprimorados como renderização de texturas 3D, utilização de *bones* que garantem um movimento mais real a personagens 3D, entre outros. Este trabalho utilizou a versão gratuita da Unity.⁴

3.2.1 Rendering

O ambiente de desenvolvimento gráfico da Unity tem a capacidade de utilizar diversas linguagens para sua renderização, sempre de acordo com a escolha da plataforma de publicação do conteúdo criado: Direct3D (Windows, Xbox 360), OpenGL (Mac, Windows, Linux, PS3), OpenGL ES (Android, iOS) e APIs proprietárias (Wii). Este ambiente conta com a capacidade de suporte para mapeamento de relevo e reflexão, sombras dinâmicas, renderização de texturas, utilização de *ambient occlusion* (característica de como a luz se irradia na vida real, principalmente em superfícies não reflexivas), entre outras. Todas estas características definem o nível de qualidade gráfica que o jogo ou simulação poderá atingir, por este motivo faz-se necessário elencá-las. Atualmente suporta a importação nativa de modelos tridimensionais dos seguintes softwares: Autodesk Maya, Autodesk 3ds Max, Cheetah 3D, Cinema 4D, Blender, Modo, Carrara Software, Lightwave, XSI, SketchUp Pro, Wings 3D, Wavefront e também para os formatos: Autodesk FBX, COLLADA, 3D Studio e DXF.

O ambiente ainda conta com a linguagem ShaderLab que é usada para a criação de *shaders* (que são conjuntos de instruções para o processamento de efeitos de renderização em uma imagem tridimensional). Ao utilizarmos um sistema desse tipo, garantimos que as imagens serão constantemente processadas, o que gera maior flexibilidade aos programadores e maior qualidade gráfica aos jogadores.

3.2.2 Scripting

A Unity conta com um ambiente de programação, chamado Scripting, construído sobre o *framework* MonoDevelop que é *open-source*. Graças a esse ambiente, a Unity possui a capacidade de interagir com as seguintes linguagens: JavaScript, C e Boo (que tem a sintaxe inspirada pela linguagem Python). A partir da versão 3.0 a Unity conta com uma versão personalizada do MonoDevelop, o que acrescentou maior versatilidade para os *scripts* dentro do ambiente de desenvolvimento. Este ambiente traz uma praticidade e uma versatilidade ao programador por

⁴(Em: <http://portuguese.unity3d.com/unity/>. Acesso em: 12 agosto 2013.)

não forçá-lo a apenas um tipo de linguagem, como por vezes acontece em outros ambientes de desenvolvimento.

3.2.3 Asset Tracking

A Unity também inclui o Asset Tracing (podendo ser traduzido como um ambiente de versionamento) que seria uma solução de controle de versão do projeto para o desenvolvedor. Este ambiente utiliza PostgreSQL como um *backend* e conta com inúmeros recursos, tais como, sistema de áudio, reprodução de vídeo entre outros.

3.2.4 Physics

A Unity também conta com o motor de física PhysX da Nvidia. Este suporte foi integrado a partir da versão 3.0 e adiciona uma melhoria nas animações que contam com detecção de colisões, manipulação de gravidade, força e torque em objetos virtuais, parâmetros de corpos rígidos, alterações de propriedades físicas de acordo com o material aplicado ao objeto e juntas (podendo assim criar objetos encadeados).

3.2.5 Versão 4.X

A Unity 4 foi anunciada em 18 de junho de 2012 e propõe o acréscimo de uma série de novas funcionalidades e reformulação de outras. Como sempre acontece as versões da Unity passam por um ciclo de vida, sendo sempre nomeadas com o número da versão, acrescido do número da revisão, por exemplo 4.1. A versão 4 teve o lançamento oficial em 14 de novembro de 2012. O estudo contido neste trabalho optou por manter utilização da versão 3.X, pois já se encontrava em fase de testes quando a nova versão foi lançada.

Os novos recursos incluem suporte para DirectX 11 e animação Mecanim, melhorias de gráficos móveis que incluem sombras em tempo real, melhorias nas características das texturas de pele, a capacidade de usar melhores mapas de textura e de luz e uma GPU de perfil refinado. Além disso, conta agora com o desenvolvimento para Adobe Flash.

3.3 Modelo Dinâmico na Unity

O ambiente de desenvolvimento Unity possibilita ao desenvolvedor fornecer características reais a objetos virtuais.

O presente trabalho utilizou-se destas características para criar um objeto virtual que simulasse e fornecesse características de força e gravidade ao sistema robótico desenvolvido para a reabilitação dos pacientes.

O modelo em questão trata-se de uma plataforma composta por uma prancha de madeira e duas manoplas manipuladoras desta prancha, conforme visto na Figura 3.4.

Figura 3.4: Modelo Virtual no ambiente de desenvolvimento Unity.

Utilizando das propriedades disponíveis pelo motor de física PhysX foi possível definir parâmetros reais para os objetos virtuais.

A Figura 3.5 apresenta uma tela de definição de parâmetros de um objeto do tipo *Rigidbody*. Nesta tela é possível identificar alguns campos de parâmetros que fornecem características reais a este objeto, tais como o campo *Mass* (massa) que fornece massa ao objeto, o campo *Use Gravity* que possibilita a ação da gravidade sobre este corpo ou não, o campo *Collision Detection* que define o tipo de colisão que este corpo rígido sofre, entre outros.

Outro campo importante que a Figura 3.5 apresenta é o *Script*. Este campo tem como ação associar uma sequência de código (*script*) ao objeto.

Como exemplo, tem-se no presente estudo o *script* Platform.cs cuja responsabilidade é calcular e transmitir a posição da prancha para o servidor de controle de comunicação, juntamente

Figura 3.5: Parâmetros de configuração do modelo virtual.

com o torque, para simular a força que o objeto virtual está recebendo no ambiente lúdico, transmitindo assim ao sistema robótico a mesma força para que o paciente tenha a exata sensação das mudanças ocorridas dentro do jogo, proporcionando uma imersão total do mesmo. O *script* utilizado neste trabalho será explicado no **Capítulo 4**.

Estas características juntamente com a propriedade que o ambiente de desenvolvimento Unity tem de possibilitar ao desenvolvedor alterar parâmetros em tempo real, ou seja, enquanto o jogo está sendo testado formam o modelo dinâmico neste ambiente de desenvolvimento.

3.4 Teleoperação Robótica

Para a implementação dos jogos cooperativos faz-se necessário o estudo de sistemas de teleoperação robótica, que permite a interação de força entre os dois sistemas robóticos utilizados nos testes. De acordo com as informações identificadas no **Capítulo 2 - Revisão Bibliográfica**, foram elencadas três estratégias de controle de sistemas teleoperados para serem aplicadas no presente trabalho: controle teleoperado direto (PD Simétrico), o controle teleoperado por variáveis de ondas (NIEMEYER (1996)) e o controle teleoperado através de ambiente virtual (CARIGNAN E OLSSON (2004)).

O conceito de sistema robótico teleoperado pode ser descrito pela Figura 3.6, que considera a teleoperação como uma cadeia de elementos transmitindo movimento e força do operador humano para um ambiente remoto ou, outro operador humano. No ambiente local, o operador

humano interage com um manipulador mestre. De forma similar, no ambiente remoto o robô manipulador escravo interage com um determinado objeto ou com outro ser humano. Os dois computadores estão conectados via Internet nos dois sentidos, de tal forma que uma malha fechada de comunicação é criada.

Figura 3.6: Sistema robótico teleoperado.

Devido aos atrasos presentes no sistema de comunicação de dados via Internet, deve-se considerar a estabilidade do sistema teleoperado. Esta análise é geralmente realizada utilizando conceitos de passividade, que prove um modo simples e robusto para analisar a estabilidade de sistemas baseada apenas nas propriedades da entrada e da saída. Por não considerar os estados internos, o critério de estabilidade baseado na passividade pode ser aplicado em uma variedade grande de sistemas, incluindo sistemas lineares, não lineares, discretos no tempo e distribuídos.

Um dado sistema é dito passivo se absorve mais energia que produz. Para sistemas mecânicos, em que a potência é definida como o produto da velocidade ($\dot{x}$) pela força (F), o critério de passividade é dado por:

$$\int_0^t P_{in} d\tau = \int_0^t \dot{x}^T F d\tau = E(t) - E(0) + \int_0^t P_{diss} d\tau, \quad (3.1)$$

sendo P_{in} a potência de entrada, $E(t) \geq 0$ a energia armazenada no instante t , $E(0) \geq 0$, o valor da energia armazenada no instante inicial e $P_{diss} \geq 0$ a potência dissipada pelo sistema. Uma análise completa da viabilidade de se utilizar a Eq. 3.1 como critério de estabilidade para sistemas teleoperados pode ser encontrada em Niemeyer (1996).

3.4.1 Método 01: Controle Teleoperado Direto (PD Simétrico)

Entre as estratégias de controle teleoperado estudadas, primeiramente considerou-se a implementação do Controle Teleoperado Direto (PD Simétrico). Considere que os dois robôs manipuladores, mestre e escravo, possam ser descritos, respectivamente, pelas equações dinâmicas:

$$M_m(q_m)\ddot{q}_m + C_m(q_m, \dot{q}_m)\dot{q}_m = -\tau_m, \quad (3.2)$$

$$M_s(q_s)\ddot{q}_s + C_s(q_s, \dot{q}_s)\dot{q}_s = \tau_s, \quad (3.3)$$

sendo os subscritos m e s referentes aos robôs mestre e escravo, respectivamente, $M(q)$ a matriz de inércia, $C(q, \dot{q})$ a matriz de forças centrípetas e Coriolis, e τ os torques aplicados nas juntas. Por simplicidade, desconsidera-se o termo gravitacional visto que este termo pode ser compensado pelo controlador. O sinal negativo na primeira equação é utilizado para manter a convenção de sinais a ser utilizada no controle por variáveis de onda.

Figura 3.7: Controle PD simétrico.

O controle PD simétrico, mostrado na Figura 3.7, é dado por:

$$\tau_m = K(q_m - q_s) + B(\dot{q}_m - \dot{q}_s), \quad (3.4)$$

$$\tau_s = K(q_m - q_s) + B(\dot{q}_m - \dot{q}_s), \quad (3.5)$$

sendo K e B ganhos constantes simétricos positivos definidos. Note que nesta configuração de controle, a posição do mestre é definida como posição desejada para o escravo, e vice-versa.

Entretanto, quando se considera o atraso decorrente da comunicação entre os robôs, os

torques aplicados aos robôs são dados por:

$$\tau_m = K(q_m(t) - q_s(t - T)) + B(\dot{q}_m(t) - \dot{q}_s(t - T)), \quad (3.6)$$

$$\tau_s = K(q_m(t - T) - q_s(t)) + B(\dot{q}_m(t - T) - \dot{q}_s(t)), \quad (3.7)$$

sendo T o atraso. Neste caso, a condição de passividade não é satisfeita e o sistema torna-se instável para valores elevados de atraso. Uma análise detalhada deste caso pode ser encontrada em Niemeyer (1996).

3.4.2 Método 02: Controle Teleoperado por Variáveis de Ondas

O segundo método estudado e implementado é o controle teleoperado por variáveis de onda. A utilização de variáveis de onda (ou transformada de onda) garante a estabilidade do sistema, mesmo na presença de valores elevados de atraso. Para o caso de sistemas robóticos teleoperados, as variáveis de onda são definidas conforme as seguintes relações:

$$u = \frac{b\dot{q} + \tau}{\sqrt{2b}}, \quad (3.8)$$

$$v = \frac{b\dot{q} - \tau}{\sqrt{2b}}, \quad (3.9)$$

sendo b a impedância da onda. As equações acima são formuladas considerando como entradas da transformada de onda as variáveis de velocidade ($\dot{q}$) e torque (τ) e como saídas u e v , definidas como variáveis de onda. Entretanto, qualquer combinação de duas variáveis de entrada e duas de saída podem ser utilizadas, como por exemplo, a transformação dada pela Figura 3.8. Neste caso as variáveis de entrada são $\dot{q}$ e v , e as variáveis de saída são u e τ . As expressões são dadas por:

$$u = \sqrt{2b}\dot{q} - v \quad (3.10)$$

$$\tau = \sqrt{2b}v + b\dot{q}. \quad (3.11)$$

Considera-se um sistema representado no domínio das variáveis de onda, como descrito pela Figura 3.9. Considera-se que um fluxo de potência positivo é dado pelas variáveis de entrada que, neste caso, é representado pela variável de onda u . A variável de onda v é tida como saída do sistema e, portanto, o fluxo de potência referente a ela é negativo. Assim, a potência inserida no sistema é dada por:

Figura 3.8: Exemplo de aplicação da transformada de onda.

Figura 3.9: Descrição do sistema no domínio das variáveis de onda.

$$P_{in} = \dot{q}^T \tau = \frac{1}{2} u^T u - \frac{1}{2} v^T v. \quad (3.12)$$

Retomando o critério de passividade dado pela Eq. 3.1, tem-se:

$$\int_0^t P_{in} d\tau = \int_0^t \frac{1}{2} u^T u - \frac{1}{2} v^T v d\tau = E(t) - E(0) + \int_0^t P_{diss} d\tau, \quad (3.13)$$

que, considerando $E(t)$ e P_{diss} positivos, gera uma condição de passividade no domínio das variáveis de onda:

$$\int_0^t \frac{1}{2} v^T v d\tau \leq \int_0^t \frac{1}{2} u^T u d\tau - E(0). \quad (3.14)$$

Como esperado, um sistema é passivo se a energia transmitida pela variável de saída (neste caso v) é limitada pela energia fornecida pela variável de entrada (neste caso u) ou pela energia inicialmente armazenada.

Este outro critério de passividade remove a dependência entre as variáveis de potência ($\dot{q}$ e τ), que na forma antiga é dada pela multiplicação entre estas variáveis. Se a variável de saída está com atraso, o efeito no produto e, como consequência, na passividade é incerto. Entretanto, no novo critério, se a variável de saída está com atraso, sua potência é temporariamente armazenada sem alterar a passividade. Como exemplo, considere que o sistema é descrito por um atraso entre as variáveis de entrada e saída, $v(t) = u(t - T)$. A condição de passividade fica:

$$\int_0^t \frac{1}{2} v^T v d\tau \leq \int_0^t \frac{1}{2} u^T u d\tau - E(0) \quad (3.15)$$

$$\Leftrightarrow \int_0^{t-T} \frac{1}{2} u^T u d\tau \leq \int_0^t \frac{1}{2} u^T u d\tau \quad (3.16)$$

$$\Leftrightarrow 0 \leq \int_{t-T}^t \frac{1}{2} u^T u d\tau, \quad (3.17)$$

que é satisfeita. A energia inicial armazenada é nula ($E(0) = 0$, assumindo que $u = 0$, para todo $t < 0$).

A representação de um sistema teleoperado básico baseado em variáveis de onda é mostrada na Figura 3.10. Os índices m e s referem-se respectivamente ao robô mestre (*master*) e ao robô escravo (*slave*). Nos dois lados, a informação sobre os robôs ($\dot{q}$ e τ) é transformada para o domínio das variáveis de onda e apenas tais variáveis são transmitidas. Considerando o atraso,

a relação entre as variáveis de onda é dada por:

$$u_s(t) = u_m(t - T), \quad (3.18)$$

$$v_m(t) = v_s(t - T), \quad (3.19)$$

sendo que as variáveis de saída (dos sistemas robóticos) são dadas por:

$$u_m(t) = \frac{b\dot{q}_m(t) + \tau_m(t)}{\sqrt{2b}}, \quad (3.20)$$

$$v_s(t) = \frac{b\dot{q}_s(t) - \tau_s(t)}{\sqrt{2b}}. \quad (3.21)$$

Figura 3.10: Sistema teleoperado básico baseado em variáveis de onda.

Pode-se notar que as variáveis de entrada u_s e v_m podem ser decodificadas de diferentes formas, de acordo com a estrutura de controle de cada um dos robôs considerados. Por exemplo, considere a estrutura mostrada na Figura 3.11. Neste caso, o robô escravo é controlado por um controlador PD, enquanto que o torque calculado a partir da variável de onda v_m é diretamente aplicado no robô. O controlador PD do robô escravo é dado por:

$$\tau_s = K(q_s^d - q_s) + B(\dot{q}_s^d - \dot{q}_s), \quad (3.22)$$

sendo a velocidade desejada decodificada a partir da variável de onda u_s :

$$\dot{q}_s^d = \frac{\sqrt{2b}u_s - \tau_s}{b}. \quad (3.23)$$

As equações (3.22) e (3.23) podem ser combinadas para se determinar a velocidade desejada

em função apenas da variável de onda juntamente com as variáveis do robô:

$$\dot{q}_s^d = \frac{\sqrt{2b}u_s + B\dot{q}_s - K(q_s^d - q_s)}{B + b}. \quad (3.24)$$

A posição desejada deve ser calculada integrando-se a velocidade desejada ao longo do tempo.

Do lado do robô mestre, o torque a ser aplicado diretamente no robô é dado por:

$$\tau_m = b\dot{q}_m - \sqrt{2b}v_m. \quad (3.25)$$

Figura 3.11: Sistema teleoperado com controlador PD no robô escravo e aplicação direta de força no mestre.

A estrutura de controle mostrada na Figura 3.11 apresenta algumas desvantagens quando implementada em um sistema real. Como pode ser observado, três malhas internas são criadas quando se utiliza as variáveis de onda. A primeira refere-se a um amortecimento aparente criado pela transformada de onda, Eq. (3.25). Este amortecimento é sentido pelo operador quando este tenta movimentar o robô mestre. A segunda malha aparece devido às reflexões de onda geradas pela transformações dos dois lados. Quando a variável de onda u_s alcança o destino, parte do sinal retorna com a variável de onda v_s de volta ao operador, Eqs. (3.22), (3.23) e (3.21). Da mesma forma, quando a variável de retorno v_m chega na transformação local, ela é refletida na variável u_m , Eqs. (3.25) e (3.20). Tais reflexões não só transmitem pouca informação, como também podem criar distúrbios que se repetem por um longo período. A terceira malha transmite o sinal de realimentação do escravo para o mestre, sendo a principal malha de controle, que deve ser preservada. A força aplicada no robô escravo, que contém a posição e velocidade

deste codificadas no erro de acompanhamento, informa ao operador se houve movimentação ou se ocorreu um contato.

Tentando eliminar os efeitos da segunda malha e preservar os efeitos das demais, em Niemeyer (1996) é proposto um sistema teleoperado com casamento de impedâncias. A estrutura deste sistema é mostrada na Figura 3.12. Os conceitos envolvidos na definição deste sistema se relacionam com as técnicas de casamento de impedâncias utilizadas em sistemas eletrônicos. Nesta nova estrutura, os dois robôs são controlados por um controlador PD e elementos de amortecimento são inseridos nas transformadas de onda e nas leis de controle. As equações finais do sistema teleoperado são dadas por:

Figura 3.12: Sistema teleoperado com casamento de impedâncias.

$$\dot{q}_s^d = \frac{\sqrt{2b}u_s - \tau_s + D_s\dot{q}_s}{b - R_s} \quad (3.26)$$

$$\tau_s = K_s(q_s^d - q_s) + B_s(\dot{q}_s^d - \dot{q}_s) - D_s\dot{q}_s, \quad (3.27)$$

$$v_s(t) = \frac{(b + R_s)\dot{q}_s^d(t) - \tau_s(t) + D_s\dot{q}_s}{\sqrt{2b}}, \quad (3.28)$$

e

$$\dot{q}_m^d = \frac{\sqrt{2b}v_m - \tau_m - D_m\dot{q}_m}{b + R_m} \quad (3.29)$$

$$\tau_m = K_m(q_m^d - q_m) + B_m(\dot{q}_m^d - \dot{q}_m) + D_m\dot{q}_m, \quad (3.30)$$

$$u_m(t) = \frac{(b - R_m)\dot{q}_m^d(t) + \tau_m(t) - D_m\dot{q}_m}{\sqrt{2b}}. \quad (3.31)$$

Para que ocorra o casamento de impedâncias, os ganhos dos controladores PD e dos elementos

de amortecimentos devem ser definidos como:

$$K_s = \lambda_s^2 M_s, \quad B_s = \lambda_s M_s, \quad R_s = b - B_s, \quad D_s = \lambda_s M_s, \quad (3.32)$$

e

$$K_m = \lambda_m^2 M_m, \quad B_m = \lambda_m M_m, \quad R_m = b - B_m, \quad D_m = \lambda_m M_m, \quad (3.33)$$

sendo λ_s e λ_m as larguras de banda para os robôs escravo e mestre, repectivamente. Como R_s e R_m devem ser positivos, a impedância da comunicação é limitada inferiormente.

Quando se aplica a estrutura de controle teleoperado descrita acima, o sistema como um todo apresenta propriedades mecânicas aparentes, tais como inércia total, amortecimento aparente, e rigidez em regime permanente. Estes parâmetros são dados em função dos parâmetros originais dos robôs mestre e escravo, da impedância de comunicação e do atraso:

$$M = M_m + bT + M_s, \quad (3.34)$$

$$B = 2b, \quad (3.35)$$

$$K^{-1} = K_m^{-1} + \left(\frac{b}{T}\right)^{-1} + K_s^{-1}. \quad (3.36)$$

Pode-se notar que com o aumento do atraso, a inércia aparente aumenta e a rigidez do sistema diminui. Por outro lado, o aumento da impedância b faz com que todos os parâmetros aumentem, devendo ser ajustado para não ter uma inércia aparente muito grande.

3.4.3 Método 03: Controle Teleoperado através de Ambiente Virtual

O terceiro método estudado e implementado consiste na construção de um ambiente virtual, no qual segundo (Carignan e Olsson (2004)) o modelo virtual de um objeto transportado é utilizado para gerar as velocidades desejadas para ambos os robôs. Neste caso, os dois robôs são considerados mestres e o objeto virtual é considerado o escravo. A Figura 3.13 mostra a representação desta estrutura de controle em um sistema robótico teleoperado, sendo M_b a massa do objeto virtual, C_b os termos não inerciais (amortecimento, efeito da gravidade), c_{s1} e c_{s2} funções relacionado os torques gerados pelos robôs com as forças aplicadas no objeto.

Na implementação desta estrutura, considera-se que a comunicação entre os robôs mestres

Figura 3.13: Estrutura de comunicação do sistema teleoperado baseado em modelo virtual.

e o modelo virtual é realizada por meios de variáveis de onda garantindo a passividade da comunicação e, portanto, a estabilidade do sistema.

A aplicabilidade de cada método, bem como sua estruturação no presente trabalho serão apresentados e discutidos no **Capítulo 4**.

As questões acerca de confiabilidade e usabilidade de cada método são demonstrados no **Capítulo 5**.

Capítulo 4

Ambiente de Jogos para Telereabilitação

Neste capítulo é descrito como foi desenvolvido o ambiente de estudos para avaliar os métodos de telereabilitação propostos.

4.1 Plataforma Virtual

Com a finalidade de promover o ambiente de testes dos métodos de Teleoperação Robótica listados no presente trabalho, bem como promover a imersão do paciente e oferecer um ambiente lúdico de apoio para a Reabilitação Robótica, foi desenvolvido um jogo utilizando o ambiente de desenvolvimento Unity.

Tal jogo consiste em um ambiente de simulação onde o paciente controla uma representação virtual da Plataforma Háptica, dispositivo este desenvolvido junto ao Laboratório de Reabilitação Robótica (ReRob) e que sua estrutura e funcionamento serão explicados logo à diante.

Tal representação virtual consiste em uma plataforma composta de uma prancha de madeira apoiada em um pedestal, esta prancha está ligada a duas manoplas responsáveis por simular a movimentação prancha em torno de seu eixo central. Esta representação virtual como já foi dito conta com alguns objetos que serão apresentados mais detalhadamente a partir deste ponto.

A prancha de madeira (Figura 4.1) foi definida como um corpo rígido, contando assim com parâmetro de colisão, ou seja, conta com um delimitador de tamanho do objeto, fazendo com que outros objetos quando alcançarem este delimitador a prancha sofra interferência externa dos

mesmos. Isso possibilita que a prancha sofra uma movimentação quando a bola (que também é um corpo rígido) cai sobre ela. Ao possibilitar isso, o jogo se torna capaz de reproduzir inclusive a questão de impedância já citada. A Figura 4.2 apresenta a tela do jogo onde é visível a prancha e a bola.

Figura 4.1: Objeto virtual de simulação da prancha.

Figura 4.2: Plataforma Virtual desenvolvida na Unity

O objeto prancha ainda conta com um *script* de controle que configura o tipo de comunicação que a prancha deverá ter restante do sistema. Este código indica quais parâmetros serão recebidos e encaminhados por ela de acordo com o tipo de Teleoperação Robótica escolhida. A Figura 4.3 apresenta o diagrama de comunicação entre o objeto virtual e seu *script* e o sistema robótico.

O *script* conta com uma estrutura de decisão do tipo *CASE* para controlar o tipo de cálculo que o posicionamento de prancha deverá utilizar e quais variáveis serão recebidas e enviadas em cada caso, de acordo com o sistema de Teleoperação Robótica.

O *script* Platform.cs verifica primeiramente se o servidor está online: se esta verificação retornar um valor positivo, o *script* então verifica se jogo e servidor estão se comunicando. Em

Figura 4.3: Uso do modelo dinâmico da Unity.

caso de resposta afirmativa para as duas verificações, ele irá identificar qual caso foi escolhido (esta escolha ocorre no início do jogo) e então acionar os parâmetros relacionados ao caso em questão. Abaixo tem-se o código que executa estas verificações de comunicação e os cálculos de cada caso de Teleoperação Robótica aplicada ao projeto.

```
void Update () {
    if (Manager.server.IsConnected){
        if (Manager.robot != null && Manager.robot.Joints.Count == 2){
            switch (activeCase)
            {
                case 1:
                    rigidbody.useGravity = false;
                    rigidbody.isKinematic = true;

                    // Motor Front
                    angleF = (float)Manager.robot.Joints[0].Position;
                    if (angleF > 60.0f * Mathf.PI / 180.0f) {
                        angleF = 60.0f * Mathf.PI / 180.0f;
                    }

                    if (angleF < - 60.0f * Mathf.PI / 180.0f) {
                        angleF = - 60.0f * Mathf.PI / 180.0f;
                    }
            }
        }
    }
}
```

```
Manager.game.Joints[0].Position = angleF * -1.0f;

// Motor Rear
angleR = (float)Manager.robot.Joints[1].Position;

if (angleR > 60.0f * Mathf.PI / 180.0f) {
angleR = 60.0f * Mathf.PI / 180.0f;
}

if (angleR < -60.0f * Mathf.PI / 180.0f) {
angleR = -60.0f * Mathf.PI / 180.0f;
}
Manager.game.Joints[1].Position = angleR * -1.0f;

// Platform
angle = ((angleF * -1.0f) + angleR)* -1.0f * 0.5f * 0.2f * 180.0f / Mathf.PI;

transform.rotation = Quaternion.AngleAxis(angle, Vector3.forward);
break;

case 2:
rigidbody.useGravity = false;
rigidbody.isKinematic = true;

// Motor Front
waveF = (float)Manager.robot.Joints[0].Wave;
Manager.game.Joints[0].Wave = waveF;

angleF = (float)Manager.robot.Joints[0].Position;
if (angleF > 60.0f * Mathf.PI / 180.0f) {
angleF = 60.0f * Mathf.PI / 180.0f;
}
}
```

```
if (angleF < - 60.0f * Mathf.PI / 180.0f) {
angleF = - 60.0f * Mathf.PI / 180.0f;
}

// Motor Rear
waveR = (float)Manager.robot.Joints[1].Wave;
Manager.game.Joints[1].Wave = waveR;

angleR = (float)Manager.robot.Joints[1].Position;

if (angleR > 60.0f * Mathf.PI / 180.0f) {
angleR = 60.0f * Mathf.PI / 180.0f;
}

if (angleR < -60.0f * Mathf.PI / 180.0f) {
angleR = -60.0f * Mathf.PI / 180.0f;
}

// Platafom
angle = ((angleF * -1.0f) + angleR)* -1.0f * 0.5f * 0.2f * 180.0f / Mathf.PI;
transform.rotation = Quaternion.AngleAxis(angle, Vector3.forward);

break;

case 3:
rigidbody.useGravity = true;
rigidbody.isKinematic = false;

// Motor Front
torqueF = (float)Manager.robot.Joints[0].Torque * -1.0f;

// Motor Rear
torqueR = (float)Manager.robot.Joints[1].Torque;
```

```

rigidbody.AddTorque (0,0,(torqueR + torqueF)/5.0f);

angle = (transform.rotation.eulerAngles.z)* Mathf.PI / 180.0f;

if (angle > Mathf.PI) {
angle  = angle - 2* Mathf.PI;
}

Manager.game.Joints[0].Position = 5.0 *angle ;
Manager.game.Joints[1].Position = 5.0 * angle * -1.0f;

break;
}
}
}
}
}

```

As manoplas são apenas representações visuais das manoplas contidas na Plataforma Háptica, não contando com nenhuma tarefa ou programação específica dentro do jogo. A Figura 4.4 apresenta as mesmas dentro do ambiente de desenvolvimento do jogo.

Figura 4.4: Objetos virtuais de simulação das manoplas.

O jogo ainda conta com um *script* denominado SpawnerBall.cs, este *script* tem como função instanciar a bola dentro do ambiente do jogo, ou seja, é ele que cria ela no início e toda vez que o jogo é reiniciado (quando a bola toca o chão o jogo é reiniciado). Tal *script* pode ser visualizado

abaixo.

```
using UnityEngine;
using System.Collections;

public class SpawnerBall : MonoBehaviour {

    public Transform ballPrefab;
    private Transform ball;
    public bool withBall = true;

    void Start()
    {
        RestartBall();
    }

    public void RestartBall()
    {
        if (ball)
            Destroy(ball.gameObject);

        if (withBall)
            ball = Instantiate(ballPrefab, transform.position, transform.rotation)
as Transform;
    }
}
```

No código acima é identificado que o *script* SpawnerBall.cs faz a chamada de um segundo *script* denominado RestartBall.cs, cuja função é recriar o objeto caso ele já tenha sido criado. Seu código pode ser visualizado abaixo.

```
using UnityEngine;
using System.Collections;
```

```
public class SpawnerBall : MonoBehaviour {

    public Transform ballPrefab;
    private Transform ball;
    public bool withBall = true;

    void Start()
    {
        RestartBall();
    }

    public void RestartBall()
    {
        if (ball)
            Destroy(ball.gameObject);

        if (withBall)
            ball = Instantiate(ballPrefab, transform.position, transform.rotation) as Transform;
    }
}
```

Tratando da questão do ambiente lúdico que o jogo é capaz de criar, tem-se que o mesmo foi desenvolvido para ser utilizado por até dois jogadores simultâneos. Como primeira possibilidade, o jogador teria de vencer ele mesmo, ou seja, sua própria lesão. Neste caso o paciente utilizaria uma das manoplas da Plataforma Háptica e através de seus próprios movimentos manter a bola equilibrada o máximo possível.

No caso de dois pacientes, cada um utilizaria uma das manoplas Plataforma Háptica e o objetivo passaria então a ser a possibilidade um paciente fazer o outro derrubar a bola, forçando movimentos em sentido contrário do seu oponente.

Sendo então seu principal objetivo o de manter a bola sobre a prancha, sem deixá-la cair, os pacientes/jogadores são expostos a um estímulo que pode desencadear uma maior motivação para praticarem a repetição dos movimentos necessários para que seja realizada a reabilitação de punho (lesão em foco neste projeto).

4.2 Plataforma Háptica

A Plataforma Háptica foi desenvolvida para ser um sistema robótico focado em reabilitação de punho, porém os métodos de Teleoperação Robótica utilizados no presente trabalho estão sendo testados para que possam ser utilizados em qualquer ambiente de Telereabilitação Robótica.

Esta plataforma é composta por uma base metálica, onde todas as outras peças estão acopladas ou fixadas. A Figura 4.5 apresenta a plataforma montada.

Figura 4.5: Plataforma Háptica

Ainda tomando como referencial a Figura 4.5 é possível verificar que os motores e as manoplas estão fixadas diretamente a base através de um dispositivo de fixação em “L”, esta fixação forneceu a estabilidade necessária ao conjunto para que os testes dos métodos de Teleoperação Robótica fossem testados a contento.

Como a Plataforma pode ser utilizada por até dois pacientes, optou-se por uma disposição onde os motores e as manoplas estivessem frente uma a outra facilitando assim sua utilização e não necessitando de montagem de acordo com a situação (um ou dois pacientes)

A estrutura que compõe a plataforma conta ainda com dois motores Maxom e uma controladora de comunicação CAN EPOS para efetuar a conexão do sistema robótico com o servidor de comunicação através de uma controladora CAN. As suas especificações não são de relevância para o presente trabalho, portanto optou-se por não citá-los.

A Figura 4.6 apresenta a controladora fixada na parte inferior da plataforma.

A comunicação da Plataforma Háptica com a Plataforma Virtual ocorre em linhas gerais da seguinte forma: os motores se comunicam através da controladora de comunicação CAN com

Figura 4.6: Detalhes da EPOS da Plataforma Háptica

o servidor de comunicação. Este por sua vez se comunica através de *sockets* e TCP/IP com o computador onde se encontra o jogo.

Quando a informação chega ao jogo e é processada, o caminho inverso começa a ser realizado, ou seja, o jogo retorna os parâmetros como posição, velocidade e variáveis de controle de ondas (caso esteja implementado) para o servidor, que retorna estas novas informações a a controlador CAN que faz as modificações necessárias na posição, velocidade e torque dos motores fechando o ciclo de comunicação das duas plataformas. A Figura 4.7 apresenta o diagrama dessa comunicação.

Figura 4.7: Sistema de comunicação da Plataforma Háptica com a Plataforma Virtual

Tratando da comunicação da controladora CAN com o servidor de comunicação, o servidor de comunicação então recebe posição, velocidade e torque dos motores e trata estas informações, colocando-as dentro de uma mensagem utilizando JSON. A mensagem encaminhada com os

dados vindos da controladora pode ser visualizada abaixo.

```
void set(const std::string& data) {
    rapidjson::Document document;
    if (!document.Parse<0>(data.c_str()).HasParseError()){
        if (document.HasMember("Type"))
            type = document["Type"].GetString();

        const Value& joint = document["Joints"];
        for (SizeType i = 0; i < joint.Size(); i++)
        {
            joints[i].torque = joint["Torque"].GetDouble();
            joints[i].position = joint["Position"].GetDouble();
            joints[i].velocity = joint["Velocity"].GetDouble();
            joints[i].wave = joint["Wave"].GetDouble();
        }
    }
}
```

Esta mensagem é encaminhada ao jogo, este então trata esta informação de acordo com o método de Teleoperação escolhido e encaminha uma mensagem de volta ao servidor, mensagem essa que mantém o mesmo formato da que havia sido recebida. Ambas as informações dependem primariamente da classe Joint, cuja função é definir e tratar os parâmetros já citados. A função pode ser visualizada abaixo.

```
class Joint {
public:
    Joint(double torque, double position, double velocity, double wave)
    : torque( torque ), position( position ), velocity( velocity ), wave (wave)
    {}

    virtual ~Joint() {}

    template <typename Writer>
```

```
void Serialize(Writer& writer) const {  
    writer.StartObject();  
  
    writer.String("Torque");  
    writer.Double(torque);  
  
    writer.String("Position");  
        writer.Double(position);  
  
    writer.String("Velocity");  
        writer.Double(velocity);  
  
    writer.String("Wave");  
        writer.Double(wave);  
  
    writer.EndObject();  
}  
  
double torque;  
double position;  
double velocity;  
double wave;  
  
};
```

4.3 Métodos de Teleoperação Robótica

Os sistemas de controle e fórmulas matemáticas aplicados a cada sistema de Teleoperação Robótica utilizado neste trabalho foram apresentados no **Capítulo 3**, os tópicos a seguir apresentarão como os mesmos métodos se comunicarão com o servidor de comunicação e o jogo.

4.3.1 Método 01: Controle Teleoperado Direto(PD Simétrico)

O primeiro método de teleoperação estudado e implementado foi o de Controle Teleoperado Direto (PD Simétrico), ou seja, um sistema de comunicação sem controle de ondas.

Este método considera que o sistema robótico e o jogo são respectivamente mestre e escravo, sendo assim o funcionamento do sistema se dá pela troca de posição entre o mestre e o escravo, fazendo com que a posição do mestre se torna a posição desejada do escravo (segmento de trajetória) e vice e versa.

O problema apresentado por este método encontra-se quando o mesmo é submetido a atrasos extremamente altos. Isto faz com que o sistema não atenda mais aos requisitos de passividade descritos no **Capítulo 3** e o sistema se torne instável. O diagrama que representa o funcionamento do sistema encontra-se na Figura 4.8.

Figura 4.8: Diagrama do Método 01 - Sem controle de Variáveis de Onda e sem Modelo Dinâmico da Unity

Este método aguarda os parâmetros de posição e velocidade vindos dos motores e devolve os mesmos parâmetros dos objetos virtuais ao servidor de comunicação, que por sua vez devolve a controladora CAN.

4.3.2 Método 02: Controle Teleoperado por Variáveis de Ondas

No segundo método, foi aplicado o Controle Teleoperado por Variáveis de Ondas junto à comunicação. Este tipo de controle fornece uma confiabilidade maior aos dados trocados entre todos os sistemas. Seu diagrama de funcionamento é visto na Figura 4.9.

Neste método, foi verificado a capacidade que o mesmo tem de, nas situações de atraso

Figura 4.9: Diagrama do Método 02 - Com controle de Variáveis de Onda e sem Modelo Dinâmico da Unity

muito grande, guardar a posição atual dos motores por alguns ciclos de comunicação, fazendo assim com que o sistema não se torne tão suscetível a instabilidades geradas por este motivo. Porém, ao se implementar diretamente o controle de variáveis de onda, criam-se três malhas de comunicação dentro do sistema, que ocasiona reflexões dentro do próprio sistema de comunicação, proporcionando a sensação que o motor oposto sofreu alguma movimentação ou até mesmo resistência em relação ao motor atual, sem que o mesmo tenha sido manipulado.

Este método aguarda os parâmetros de posição e variáveis de ondas vindos dos motores e devolve apenas o cálculo de retorno das variáveis de onda ao servidor de comunicação, que por sua vez encontra a nova posição dos motores e devolve esta informação a controladora CAN.

4.3.3 Método 03: Controle Teleoperado através de Ambiente Virtual

O último método estudado utiliza Controle Teleoperado através de Ambiente Virtual, ou seja, os objetos virtuais contidos dentro do ambiente de desenvolvimento Unity, simulam parâmetros do mundo real, como massa e gravidade e esses parâmetros são acrescentados a todo o controle de comunicação.

Este método em questão não utilizou do controle de variáveis apresentado junto ao método de Controle Teleoperado por Variáveis de Ondas, portanto o robô é o mestre e o jogo é o escravo. A Figura 4.10 descreve o diagrama de funcionamento deste método.

Essa utilização proporciona a seguinte situação: o posicionamento da manopla na plataforma háptica se torna exatamente o mesmo que o da plataforma virtual, ou seja, quando a prancha

Figura 4.10: Diagrama do Método 03 - Sem controle de Variáveis de Onda e com Modelo Dinâmico da Unity

de madeira que compõe a plataforma virtual colide com o chão, estas propriedades aplicadas ao objeto virtual fazem com que a prancha não ultrapasse o limite do chão, e isso reflete no limite máximo de giro da manopla na plataforma háptica. Este método se torna extremamente eficaz quando visualizado pela ótica da imersão do paciente junto ao ambiente lúdico.

Este método aguarda os parâmetros de torque vindo dos motores e devolve os parâmetros de posição e velocidade ao servidor de comunicação, que por sua vez devolve esta informação a controladora CAN.

Capítulo 5

Resultados

Neste capítulo são apresentados os resultados experimentais obtidos através das análises dos métodos de telerabilitação propostos. Todos os modelos foram submetidos aos mesmos testes, resultando em dados de comparação confiáveis e fiéis.

Com base nos dados e trabalhos levantados na Revisão Bibliográfica apresentada no **Capítulo 2**, o presente trabalho submeteu todos os métodos de Teleoperação Robótica a testes que envolvem atrasos artificiais, em decorrência de todos os sistemas (Robô, Servidor de Comunicação e Jogo) encontrarem-se na mesma rede.

Ainda de acordo com dados obtidos através da Revisão Bibliográfica, foram aplicadas 4 possibilidades de atraso artificiais todos em milissegundos para simular a comunicação via Internet. Os valores aplicados para estas simulações são os seguintes:

- Atraso de 1 milissegundo para simular uma rede com o mesmo quase nulo;
- Atraso de 300 milissegundos para simular um tráfego baixo na rede;
- Atraso de 600 milissegundos para simular um tráfego mediano na rede;
- Atraso de 900 milissegundos para simular um tráfego alto na rede.

Os resultados destes testes encontram-se elencados abaixo.

5.1 Método 01: Controle Teleoperado Direto (PD Simétrico)

Ao submetermos o método de Controle Teleoperado Direto (PD Simétrico) ao teste de atraso igual a 1, pode-se identificar que o atraso existente (aproximadamente 2 milissegundos) é relativamente baixo quando relacionamos a posição do motor 02 com a posição do motor 01, conforme evidenciado na Figura 5.1, não trazendo prejuízo a imersão do paciente. Como já explicado nos **Capítulos 3 e 4**, ambos os motores contam com as mesmas características, portanto com a finalidade de gerar uma identificação para facilitar o entendimento dos gráficos abaixo, o motor 01 será o correspondente à manopla frontal da Plataforma Virtual.

Figura 5.1: Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 01.

Nas Figuras 5.2 e 5.4 podem ser encontradas as relações de Posição x Tempo e Torque x Tempo respectivamente.

Figura 5.2: Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 01.

Figura 5.3: Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 01.

Na análise, pode-se verificar que a posição em relação ao tempo de resposta continua numa condição de atraso quase nulo, podendo ser desprezado. Porém ao identificar a relação de Torque x Tempo, verifica-se uma condição desfavorável do torque exercido pelo Motor 02 em relação ao Motor 01, o que poderá ocasionar uma certa instabilidade ao sistema. Isso fica evidenciado também pela Figura 5.3, ao verificar as velocidades dos dois motores nesse método.

Figura 5.4: Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 01.

Ao aplicar o atraso artificial igual a 300 milissegundos, o segmento de trajetória entre os motores 1 e 2 sofre de falta de sincronismo no posicionamento angular de cada um deles, entretanto com relação ao tempo, existe um sincronismo duradouro, de acordo com as Figuras 5.5 e 5.6.

Figura 5.5: Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 300.

O sincronismo entre o torque dos motores ainda se mantém preservado nesse caso, conforme a Figura 5.7 demonstra. Com relação a parâmetro de velocidade, foi identificado que já começa a existir uma falha de sincronismo entre os motores da plataforma, conforme a Figura 5.8.

Figura 5.6: Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 300.

Figura 5.7: Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 300.

Figura 5.8: Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 300.

Quando se aplica o atraso artificial igual a 600, encontra-se a um atraso real entre o sincronismo dos motores próximo de 8 milissegundos, bem como o sincronismo dos torques. O parâmetro de velocidade sofre o mesmo problema, mostrando a falta de capacidade do método de manter o sincronismo do sistema durante altos índices de atraso artificial. As Figuras 5.9, 5.10 e 5.12 evidenciam isso graficamente, juntamente com a Figura 5.11.

Figura 5.9: Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 600.

Figura 5.10: Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 600.

Figura 5.11: Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 600.

Figura 5.12: Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 600.

Ao aplicar um atraso igual a 900, o sistema acaba por apresentar instabilidade, fazendo com que o sincronismo dos dois motores seja perdido permanentemente após atingir a marca de 80 segundos, conforme fica evidenciado na Figura 5.13. Como consequência toda a comunicação do método de telereabilitação, assim com o jogo, torna-se completamente nulo.

Figura 5.13: Posição Atual x Posição Desejada dos Motores - Método 01 com atraso igual a 900.

Ao analisar as Figuras 5.14 e 5.15 percebe-se claramente o prejuízo causado na perda de sincronismo dos Motores, ocasionando inclusive a falta de sincronismo no torque e na velocidade (Figura 5.16), o que acarreta uma falta da sensação de “peso” exercido pelo modelo virtual do jogo, prejudicando desta forma toda a estrutura lúdica da reabilitação do paciente.

Figura 5.14: Posição dos Motores em relação ao tempo de resposta - Método 01 com atraso igual a 900.

Figura 5.15: Relação de Torque x Tempo entre os dois motores - Método 01 com atraso igual a 900.

Figura 5.16: Velocidade dos Motores relacionada ao tempo - Método 01 com atraso igual a 900.

5.2 Método 02: Controle Teleoperado por Variáveis de Ondas

Este método, em relação ao anterior traz como melhoria a implementação de Variáveis de Onda e, ao ser submetido aos testes, encontram-se os seguintes resultados:

No teste de atraso igual a 1, tem-se uma relação das posições dos motores muito próxima do ideal, conforme fica evidenciado na Figura 5.17.

Figura 5.17: Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 01.

O Controle Teleoperado por Variáveis de Onda proporciona uma estabilidade na entrega dos dados de posição e torque através da rede e, conseqüentemente, tem-se uma troca de posição entre os motores (segmento de trajetória), muito mais próxima de um padrão ideal (atraso quase 0), conforme demonstrado na Figura 5.18.

Figura 5.18: Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 01.

Essa estabilidade é refletida consideravelmente na relação do torque dos motores, como pode-se ver nos gráficos demonstrados pelas Figuras 5.19 e 5.20. Isso auxilia o paciente a conseguir uma maior imersão dentro do ambiente lúdico do jogo.

Figura 5.19: Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 01.

Figura 5.20: Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 01.

Quando se submete esse método ao atraso de 300 milissegundos percebe-se um início na diferença da posição dos motores, elevando o atraso da posição atual para a posição desejada em aproximadamente 3 milissegundos. Nas Figuras 5.21, 5.22 e 5.23 identifica-se esse resultado através das comparações das posições e do torque dos motores. Analisando a Figura 5.24 tem-se o atraso se formando também na comparação do parâmetro de velocidade. Isso demonstra que quando submetido a atrasos grandes da rede de comunicação, o método baseado no Controle Teleoperado por Variáveis de Ondas é capaz de minimizar esse atraso nos objetos reais (motores).

Figura 5.21: Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 300.

Figura 5.22: Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 300.

Figura 5.23: Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 300.

Figura 5.24: Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 300.

Conforme a leitura, é possível que seja teorizada a afirmação de que quanto maior é o atraso artificial que se aplica à rede, maior deveria ser tendência em existir diferença entre a posição do motor 1 para o motor 2.

A implementação desse método, busca refutar esta afirmação e ao se analisar as Figuras 5.25, 5.26 e 5.27, a afirmação realmente se torna falha.

Figura 5.25: Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 600.

Figura 5.26: Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 600.

Figura 5.27: Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 600.

Estas Figuras demonstram através de gráficos que, mesmo com um atraso artificial grande na rede, as posições dos motores não se alteram significativamente, causando apenas um retardamento de movimentação dos motores (seguimento de trajetórias), mas mantendo o sistema estável. Existe também uma falta de sincronismo com relação as velocidades dos motores em acréscimo quando comparada aos dois primeiros casos de atraso artificial, conforme mostra a Figura 5.28.

Figura 5.28: Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 600.

Com um atraso artificial de 900 milissegundos aplicado ao sistema, tem-se um acréscimo do atraso real de aproximadamente 7 milissegundo, ao acompanhar o segmento de trajetória entre os dois motores, conforme a Figura 5.29 demonstra esta afirmação. Esse atraso ainda é baixo se considerado o alto valor de atraso artificial aplicado. A Figura 5.30 mostra o atraso entre os motores.

Figura 5.29: Posição Atual x Posição Desejada dos Motores - Método 02 com atraso igual a 900.

Figura 5.30: Posição dos Motores em relação ao tempo de resposta - Método 02 com atraso igual a 900.

Já com relação ao torque, pode-se verificar que devido ao atraso, o motor 02 não consegue exercer todo o parâmetro de torque fornecido pelo motor 01 conforme visto na Figura 5.31.

Figura 5.31: Relação de Torque x Tempo entre os dois motores - Método 02 com atraso igual a 900.

Essa falha também altera a percepção do paciente em relação ao ambiente lúdico do jogo, prejudicando sua reabilitação. As velocidades também não são exatamente iguais em ambos os motores, possibilitando que uma manopla se movimente mais rápido que a manopla oposta (Figura 5.32).

Figura 5.32: Velocidade dos Motores relacionada ao tempo - Método 02 com atraso igual a 900.

5.3 Método 03: Controle Teleoperado através de Ambiente Virtual

Ao implementar o método que utiliza o objeto virtual como escravo, identifica-se a maior divergência de posicionamento de todos os experimentos. Em nenhum dos testes com os atrasos artificiais um motor conseguiu seguir o caminho do outro de forma eficaz, entretanto não houve instabilidade do sistema.

Quando aplica-se o atraso igual a 1, pode-se identificar que a posição do motor 2 não segue a posição do motor 1 conforme desejado, de acordo com as Figuras 5.33 e 5.34, já alcançando um atraso real de quase seis milissegundos.

Figura 5.33: Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 01.

Figura 5.34: Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 01.

Verifica-se também nas Figuras 5.35 e 5.36 que os torques e as velocidades estiveram fora de sincronismo durante o experimento.

Figura 5.35: Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 01.

Figura 5.36: Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 01.

Durante o teste de atraso artificial de 300 milissegundos, o atraso no posicionamento da trajetória dos motores não sofreu aumento significativo em relação ao teste anterior, porém, os motores continuam incapazes de acompanhar a movimentação um do outro dentro de um limite de atraso que não cause percepção ao paciente. Este limite de atraso não pode ser expresso em números, por conta do grau da lesão apresentada pelo paciente, portanto o presente trabalho concentra-se apenas no grau de percepção visual e tátil do paciente.

Vale ressaltar que o sistema mantém-se estável, conforme evidenciado na Figura 5.37 e mais claramente na Figura 5.38.

Figura 5.37: Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 300.

Figura 5.38: Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 300.

Figura 5.39: Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 300.

Os parâmetros de torque e velocidade sofrem as mesmas consequências de falha de sincronismo, conforme Figuras 5.39 e 5.40.

Figura 5.40: Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 300.

Ao implementar o atraso artificial de 600 milissegundos, verifica-se nas Figuras 5.41 e 5.42, que o motor 2 não possui mais capacidade alguma de seguir a posição do motor 1. Essa situação ocorre devido a demora da entrega de informação entre o servidor de controle da plataforma háptica e o computador onde o jogo é executado. O atraso real chega a aproximadamente 8 milissegundos.

Figura 5.41: Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 600.

Figura 5.42: Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 600.

O mesmo ocorre com os parâmetros de torque e velocidade durante o atraso artificial de 600, conforme Figuras 5.43 e 5.44.

Figura 5.43: Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 600.

Figura 5.44: Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 600.

Ao implementar o atraso de 900 milissegundos, o método em questão apresentou a maior diferença entre o posicionamento dos motores, juntamente com o maior índice de atraso real de todos os teste, chegando a aproximadamente 12 milissegundos. Esse dado pode ser comprovado nas Figuras 5.45 e 5.46,

Figura 5.45: Posição Atual x Posição Desejada dos Motores - Método 03 com atraso igual a 900.

Figura 5.46: Posição dos Motores em relação ao tempo de resposta - Método 03 com atraso igual a 900.

Os parâmetros de torque e velocidade, evidentes nas Figuras 5.47 e 5.48 também apresentam os maiores índices de atraso.

Figura 5.47: Relação de Torque x Tempo entre os dois motores - Método 03 com atraso igual a 900.

Figura 5.48: Velocidade dos Motores relacionada ao tempo - Método 03 com atraso igual a 900.

A informação que deve ser ressaltada é que mesmo com o maior índice de atraso real de todo o experimento, este método não se tornou instável.

Os presentes dados foram publicados junto ao IEEE 3rd International Conference on Serious Games and Applications for Heal (SeGAH 2014) sob o título Robotic Platform for Telerehabilitation Studies based on Unity Game Engine

5.4 Tabela Comparativa

Para facilitar a identificação das principais diferenças entre os métodos, criou-se a Tabela 5.1. Esta tabela elenca os principais pontos e falhas de cada método.

O método de comunicação baseado em um Controle Teleoperado Direto (PD Simétrico) mostrou-se extremamente estável a índices de atrasos baixos (até 300 milissegundos), mas ineficaz quando submetido aos altos índices de atraso (600 e 900 milissegundos), gerando assim falha de sincronismo entre a trajetória dos dois motores, ou seja, o motor 2 não consegue mais seguir a posição do motor 1.

O método de comunicação baseado em Controle Teleoperado por Variáveis de Ondas, apresentou um sincronismo de posicionamento dos motores muito próximo do ideal, que seria um valor de atraso real igual a zero. Quando submetido ao atraso artificial de 900 milissegundos, o sincronismo de posicionamento atingiu um atraso real próximo a 7 milissegundos. Comparado ao método anterior, este método acrescenta uma melhoria significativa ao sistema, pois fornece uma confiabilidade no posicionamento dos motores e uma estabilidade ao sistema.

O método de comunicação baseado em Controle Teleoperado através de Ambiente Virtual, apresentou o maior índice de atraso entre o seguimento de trajetória dos motores. Na análise dos gráficos, identifica-se que o atraso do seguimento de trajetórias entre os dois motores quando expostos à comunicações com atraso artificial inicial de 1 milissegundo, inicia-se próximo de 6 milissegundos, chegando a 11 milissegundos quando submetido ao maior índice de atraso artificial. Esse atraso elevado gera uma grande divergência entre o torque e as velocidades dos motores, porém o sistema ainda se mantém estável.

Tabela 5.1: Tabela comparativa dos métodos de teleoperação robótica.

Método	Atrasos (<i>ms</i>)			
	1	300	600	900
Método 1	Baixo atraso, estabilidade no sistema (sincronismo entre os motores)	Falta de sincronismo angular entre os motores	Atraso real de 8 milissegundo gerando instabilidade em todos os parâmetros	Sincronismo é perdido de forma permanente após 80s de teste. Falha completa no sincronismo do sistema
Método 2	Relação de posição dos motores próximo do ideal	Atraso real de aproximadamente 3 milissegundos. Sistema sincronizado e estável	Atraso na seguimento de trajetórias entre os motores, contudo o sistema continua estável	Atraso real de 7 milissegundos. Sistema perde a capacidade de sincronismo total entre os torques.
Método 3	Atraso real de 6 milissegundos. Os motores não seguem a posição corretamente, contudo o sistema é estável	Atraso real não sofreu aumento significativo em função ao teste anterior. Os motores não seguem a posição corretamente, contudo o sistema é estável	Devido a demora na entrega da informação entre o servidor e o jogo, o motor 2 não é mais capaz de seguir a trajetória do motor 1. O atraso real chega a 8 milissegundo, contudo o sistema ainda é estável	Maior índice de atraso real (12 milissegundos) e maior diferença entre o posicionamento dos motores. Entretanto, mesmo com o maior índice de atraso o sistema se manteve estável.

Capítulo 6

Conclusão

Este trabalho teve como objetivo desenvolver um ambiente de estudos para Telereabilitação e jogos cooperativos se valendo de comunicação TCP/IP e do ambiente de desenvolvimento de jogos Unity.

Utilizando o protótipo baseado na infraestrutura proposta, foram realizados testes que permitiram chegar às seguintes conclusões:

- Ao implementar um método de comunicação baseado em um Controle Teleoperado Direto (PD Simétrico) e sem nenhum uso de controle de variáveis de ondas, verificou-se que o mesmo apresenta uma ótima resposta quando exposto a atrasos baixos, conforme dados apresentados no **Capítulo 5**. O mesmo não pode ser afirmado quando o método recebe altos índices de atraso; nas avaliações com atrasos artificiais propostos de 600 e 900 milissegundos respectivamente o método apresenta falta de sincronismo nas posições dos motores (seguimento de trajetória entre ambos) e também instabilidade entre os mesmos;
- Os testes aplicados ao método de comunicação baseado em Controle Teleoperado por Variáveis de Ondas, apresentaram um sincronismo de posicionamento dos motores muito próximo de zero, tendo em vista que, com atrasos de 900 milissegundos, esse sincronismo de posicionamento dos motores atingiu um atraso máximo próximo de 7 milissegundos. O que demonstra melhoria significativa ao ser comparado com o método de Controle Teleoperado Direto (PD Simétrico), pois os motores seguem a trajetória um dos outros com um atraso relativamente baixo, não gerando instabilidade no sistema;
- O terceiro método implementado a ser submetido a teste foi o método Controle Teleoperado

através de Ambiente Virtual. Esse método mostrou-se com o maior índice de atraso entre o seguimento de trajetória dos motores; na análise dos gráfico demonstrados no **Capítulo 05**, identifica-se que o atraso do seguimento de trajetórias entre os dois motores quando expostos a comunicações com atraso artificial inicial de 1 milissegundo, inicia-se próximo de seis milissegundos. Ao analisar o maior índice de atraso artificial implementado (900 milissegundos) nos testes, identificou-se o atraso real entre as posições dos motores aproximando-se de 11 milissegundos. Esse atraso confere ao sistema divergência entre o torque dos motores e também as velocidades dos motores se comparadas entre si, mas ainda mantém a estabilidade.

6.1 Pesquisas Futuras

Ao finalizar este trabalho, algumas questões importantes, capazes de nortear pesquisas futuras devem ser colocadas:

- Verificar qual é o impacto do atraso em uma comunicação via Internet a ponto de desestabilizar o sincronismo entre o acompanhamento das trajetórias de cada motor e a estabilidade desses, fazendo com que a Plataforma Háptica acabe por perder sua função durante a reabilitação;
- Comprovar a eficácia de jogos cooperativos (jogador vs. jogador) serem capazes de tornar a Reabilitação Robótica mais produtiva em relação a procedimentos de Reabilitação Robótica baseados em jogos individuais (jogador vs. máquina);
- Averiguar a possibilidade de implementação de algoritmos adaptativos para a Telereabilitação Robótica interativa e cooperativa, gerando talvez um controle de comunicação mais robusto e ao mesmo tempo mais flexível, buscando alcançar assim que o sistema robótico não sofra atrasos que possam comprometer parâmetros de posicionamento, torque e velocidade e garanta ao mesmo tempo a confiabilidade dos parâmetros do objeto virtual (jogos), a fim de manter um sincronismo ideal entre o sistema robótico e o jogo.

Referências Bibliográficas

- Abramczuk, B. e E. Villela (2009). A luta contra o AVC no Brasil. *Com Ciência - Revista Eletrônica de Jornalismo Científico*, <http://www.comciencia.br/comciencia/> (14 de abril de 2011) 109.
- Ahn, J. e N. Hogan (2010). Feasibility of dynamic entrainment with ankle mechanical perturbation to treat locomotor deficit. In *Proceedings of 32nd Annual International Conference of the IEEE EMBS*, Buenos Aires, Argentina.
- Anderson, R. J. e M. W. Spong (1989). Bilateral control of teleoperators with time delay. *IEEE Transactions on Automatic Control* 34(5), 494–501.
- Association, A. H. (2011). *Heart Disease and Stroke Statistics 2011 Update: A Report From the American Heart Association*. Circulation - Journal of the American Heart Association.
- Benseñor, I. e P. Lotufo (2008). *A incidência do acidente vascular cerebral no Brasil*. Publicado em 10 de junho de 2008 (atualizado em 08 de dezembro de 2008) <http://saude.hsw.uol.com.br/avc-epidemiologia.htm> (14 de abril de 2011).
- Carignan, C. e P. A. Olsson (2004). Cooperative control of virtual objects over the internet using force-reflecting master arms. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, New Orleans, USA.
- Carignan, C. R. e H. I. Krebs (2006). Telerehabilitation robotics: Bright lights, big future? *Journal of Rehabilitation Research & Development* 43(4), 695–710.
- Dam, M., P. Tonin, S. Casson, M. E. G. Pizzolato, V. Iaia, e L. Battistin (1993). The effects of long-term rehabilitation therapy on post-stroke hemiplegic patients. *Stroke* 24(8), 1186–1191.
- de Geografia e Estatística (IBGE), I. B. (2010). *Sinopse do Censo Demográfico 2010*. <http://www.ibge.gov.br/home/estatistica/populacao/censo2010> (1 de maio de 2011).

- Dewald, J. P. A., P. S. Pope, J. D. Given, T. S. Buchanan, e W. Z. Rymer (1995). Abnormal muscle coactivation patterns during isometric torque generation at the elbow and shoulder in hemiparetic subjects. *Brain* 118, 495–510.
- Fasoli, S. D., H. I. Krebs, J. Stein, W. R. Frontera, e N. Hogan (2003). Effects of robotic therapy on motor impairment and recovery in chronic stroke. *Archives of Physical Medicine and Rehabilitation* 84, 477–482.
- Forrester, L. W., A. Roy, H. I. Krebs, e R. F. Macko (2011). Ankle training with a robotic device improves hemiparetic gait after a stroke. *Neurorehabilitation and Neural Repair* 25(4), 369–377.
- Gandevia, S. C. (1993). *Strength Changes in Hemiparesis: Measurements and Mechanisms*. In A.F. Thilmann, D. J. Burke, W. Z. Rymer (Eds.), *Spasticity: Mechanisms and Management* (pp. 111-122). Berlin: Springer-Verlag.
- Harwin, W. S., R. Loureiro, F. A. L. M. Taylor, G. Johnson, E. Stokes, S. Coote, M. Topping, C. Collin, S. Tamparis, J. Kontoulis, M. Munih, P. Hawkins, e B. Driessen (2001). The gentles/s project: A new method of delivering neuro-rehabilitation. *Assistive Technology-Added value to the quality of life, IOS Press*. 10, 36–41.
- Hidler, J. M., D. Nichols, M. Pelliccio, K. Brady, D. D. Campbell, J. H. Kahn, e G. T. Hornby (2009). Multicenter randomized clinical trial evaluating the effectiveness of the Lokomat in subacute stroke. *Neurorehabilitation and Neural Repair* 23, 5–13.
- Hogan, N. (1985). Impedance control: an approach to manipulation. *Journal of Dynamic Systems Measure Control* 107(Parts 1-3), 1–24.
- Hornby, G. T., D. D. Campbell, J. H. Kahn, T. Demott, J. L. Moore, e H. R. Roth (2008). Enhanced gait-related improvements after therapist versus robotic-assisted locomotor training in subjects with chronic stroke. *Stroke* 39, 1786–1792.
- Jezernik, S., G. Colombo, T. Keller, H. Frueh, e M. Morari (2003). Robotic orthosis Lokomat: A research and rehabilitation tool. *Neuromodulation* 6(2), 108–115.
- Jezernik, S., G. Colombo, e M. Morari (2004). Automatic gait-pattern adaptation algorithms for rehabilitation with a 4-dof robotic orthosis. *IEEE Transactions on Robotics and Automation* 20(3), 574–582.

-
- Krebs, H. I., L. Dipietro, S. Levy-Tzedek, S. E. Fasoli, A. Rykman-Berland, J. Zipse, J. A. Fawcett, J. Stein, H. Poizner, A. C. Lo, B. T. Volpe, e N. Hogan (2008). A paradigm shift for rehabilitation robotics: Therapeutic robots enhance clinician productivity in facilitating patient recovery. *IEEE Engineering in Medicine and Biology Magazine* 27(4), 61–70.
- Krebs, H. I., B. T. Volpe, M. L. Aisen, W. Hening, S. Adamovich, H. Poizner, K. Subrahmanyam, e N. Hogan (2003). Robotic applications in neuromotor rehabilitation. *Robotica* 21, 3–11.
- Lawrence, D. A. (1988). Impedance control stability properties in common implementations. In *Proceedings of IEEE Conference on Robotics and Automation*, Philadelphia, USA.
- Lum, P. S., S. L. Lehman, e D. J. Reinkensmeyer (1995). The bimanual lifting rehabilitator: an adaptive machine for therapy of stroke patients. *IEEE Transactions on Rehabilitation Engineering* 3, 166–174.
- Niemeyer, G. (1996). *Using Wave Variables in Time Delayed Force Reflecting Teleoperation*. Boston, USA: Ph.D Thesis - Massachusetts Institute of Technology.
- Niemeyer, G. e J. J. Slotine (1997a). Designing force reflecting teleoperators with large time delays to appear as virtual tools. In *Proceedings of the 1997 IEEE International Conference on Robotics & Automation*, Albuquerque, USA.
- Niemeyer, G. e J. J. Slotine (1997b). Using wave variables for system analysis and robot control. In *Proceedings of the 1997 IEEE International Conference on Robotics & Automation*, Albuquerque, USA.
- Niemeyer, G. e J. J. Slotine (1998). Towards force-reflecting teleoperation over the internet. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, Leuven, Belgium.
- Nudo, R. J. e K. M. Friel (1999). Cortical plasticity after stroke: implications for rehabilitation. *Revue Neurologique* 155, 713–717.
- Olsson, P. A., C. Carignan, e J. Tang (2004). Cooperative control of virtual objects using haptic teleoperation over the internet. In *Proceedings of the 5th Intl Conf. Disability, Virtual Reality & Assoc. Tech.*, Oxford, UK.
- Riener, R., L. Li¹/₂nenburger, S. Jezernik, M. Anderschitz, G. Colombo, e V. Dietz. (2005). Patient-cooperative strategies for robot-aided treadmill training: First experimental results. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 13(2), 380–394.

- Roy, A., H. I. Krebs, D. J. Williams, C. T. Bever, L. W. Forrester, R. M. Macko, e N. Hogan (2009). Robot-aided neurorehabilitation: A novel robot for ankle rehabilitation. *IEEE Transactions on Robotics* 25(3), 569–581.
- Rybak, I. A., K. Stecina, N. A. Shevtsova, e D. A. McCrea (2006). Modelling spinal circuitry involved in locomotor pattern generation: insights from the effects of afferent stimulation. *Journal of Physiology* 577, 641–658.
- Staines, W. R., W. E. McIlroy, S. J. Graham, e S. E. Black (2001). Bilateral movement enhances ipsilesional cortical activity in acute stroke: A pilot functional MRI study. *Neurology* 56, 401–404.
- Tang, J., C. Carignan, e S. Gattewar (2005). Virtual environment for robotic tele-rehabilitation. In *Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics*, Chicago, USA.
- Taub, E. e T. Elbert (2002). New treatments in neurorehabilitation founded on basic research. *Nature Reviews Neuroscience* 3(3), 228–236.
- Thilmann, A., D. J. Burke, e W. Z. Rymer (1993). *Spasticity: Mechanisms and Management*. (Eds.) Berlin: Springer-Verlag.
- Trombly, C. A. (2002). *Conceptual Foundations for Practice*. In C.A. Trombly, M. V. Radomski (Eds.), *Occupational therapy for physical dysfunction* (5th ed., pp. 1-15). Baltimore: Lippincott Williams & Wilkins.
- van der Lee, J. H., R. C. Wagenaar, G. J. Lankhorst, T. W. Vogelaar, W. L. Deville, e L. M. Bouter (1999). Forced use of the upper extremity in chronic stroke patients: results from a single-blind randomized clinical trial. *Stroke* 30, 2369–2375.
- Whiteall, J., S. McCombe-Waller, K. H. Silver, e R. F. Macko (2000). Repetitive bilateral arm training with rhythmic auditory cueing improves motor function in chronic hemiparetic stroke. *Stroke* 31, 2390–2395.