UnB - UNIVERSIDADE DE BRASÍLIA

FGA - FACULDADE GAMA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

CONTROLE DE PRÓTESES TRANSFEMURAIS ATIVAS COM *FUZZY CEREBELLAR MODEL ARTICULATION CONTROLLER*

ROBERTO AGUIAR LIMA

ORIENTADORA: Dra. LOURDES MATTOS BRASIL

CO-ORIENTADORA: Dra. VERA REGINA DA SILVA MARÃES

QUALIFICAÇÃO DE MESTRADO EM ENGENHARIA BIOMÉDICA

BRASÍLIA/DF: DEZEMBRO – 2014

UnB - UNIVERSIDADE DE BRASÍLIA

FGA - FACULDADE GAMA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

CONTROLE DE PRÓTESES TRANSFEMURAIS ATIVAS COM FUZZY CEREBELLAR MODEL ARTICULATION CONTROLLER

ROBERTO AGUIAR LIMA

QUALIFICAÇÃO DE MESTRADO EM ENGENHARIA BIOMÉDICA

APROVADA POR:

_____

Profa. Dra. Lourdes Mattos Brasil
(Orientadora)

_____

Profa. Dra. Vera Regina Fernandes da Silva Marães
(Co-orientadora)

_____

Prof. Dr. Gerardo Antonio Idrobo Pizo
(Examinador Interno ou Externo)

BRASÍLIA/DF, 08, DEZEMBRO, 2014.

RESUMO

CONTROLE DE PRÓTESES TRANSFEMURAIS ATIVAS COM *FUZZY CEREBELLAR MODEL ARTICULATION CONTROLLER*

Autor: Roberto Aguiar Lima
Orientadora: Profa. Dra. Lourdes Mattos Brasil
Co-orientadora: Dra. Vera Regina Fernandes da Silva Marães
Programa de Pós-Graduação em Engenharia Biomédica – Qualificação de Mestrado
Brasília, Dezembro de 2014.

Próteses transfemurais podem ser classificadas como ativas e passivas. As próteses transfemurais passivas têm a desvantagem de exigir do paciente um maior gasto energético metabólico. Devido a este problema, foram desenvolvidas as próteses transfemurais ativas. Estas injetam energia nos atuadores da prótese, minimizando o problema. Para que próteses transfemurais ativas possam funcionar, é necessária a criação de sistemas de controle dos seus atuadores. Estes sistemas são difíceis de projetar devido às características cinéticas e cinemáticas da marcha humana. Uma forma mais fácil de projetar estes sistemas é utilizando-se de técnicas de aprendizado de máquina. O aprendizado de máquina torna possível a criação de sistemas de controle, simplesmente escrevendo e treinando um software, a partir de dados coletados anteriormente, para que ele possa agir como um ou mais sinais de saída. As próteses transfemurais ativas requerem a criação de sistemas embarcados para que possam ser funcionais. Estes sistemas geralmente exigem baixo consumo de energia e outros recursos. Dentre as inúmeras possibilidades de algoritmos de aprendizado de máquina está o CMAC (*Cerebellar Model Articulation Controller*). Este algoritmo tem como vantagem a necessidade de se programar apenas uma tabela de *lookup* e cálculos simples para processar sua saída. Isto constitui uma característica desejável para uma prótese transfemural ativa. O objeto do presente trabalho é definir simulações que demonstrem a eficácia de uma Rede Neural Artificial (RNA) do tipo CMAC, para aproximar sinais de um joelho sadio, durante o ciclo de marcha confortável, numa prótese transfemural ativa. Além disso, o trabalho também procura a aproximação de outros tipos de ciclo de marchar usando sistemas *fuzzy* para alterar a saída da RNA CMAC.

Palavras-chaves: CMAC, prótese transfemural ativa, aprendizado de máquina.

ABSTRACT

TRANSFEMURAL PROSTHESIS CONTROL WITH FUZZY CEREBELAR MODEL ARTICULATION CONTROL

Author: Roberto Aguiar Lima
Supervisor: Dra. Lourdes Mattos Brasil
Co-supervisor: Dr(a).
Post-Graduation Program in Biomedical Engineering – Qualify of Master Degree
Brasília, December of 2014.

Transfemoral prosthesis can be classified as active and passive. Passive transfemoral prostheses have the disadvantage of requiring a higher metabolic energy expenditure of the patient. Due to this problem, actives transfemoral prosthesis were developed. These inject energy in the prosthesis actuators, minimizing the problem. For active transfemoral prosthesis to function, it is required to create control system for his actuators. These systems are difficult to design due to the kinetic and kinematic characteristics of the human gait. An easier way to design these systems is using machine learning techniques. The machine learning makes it possible to create control systems by simply writing and training software, from previously collected data, so it can act as one or more output signals. Active transfemoral prostheses require the creation of embedded systems in order to be functional. These systems often require low power consumption and other resources. Among the numerous possibilities of machine learning algorithms is the CMAC (Cerebellar Model Articulation Controller). This algorithm has the advantage of the need to program only one lookup table and simple calculations to process its output. This is a desirable feature for an active transfemoral prosthesis. The object of our study is to define simulations that demonstrate the effectiveness of an Artificial Neural Network (ANN) of the CMAC type, to aproximate the signs of a healthy knee, during the comfortable gait cycle, in a active transfemoral prosthesis. In addition, the study also tries the aproximation of other gaits cycles using fuzzy systems to modify the output of the ANN CMAC.

Key-words: CMAC, transfemural prosthesis, machine learning.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

ACM - *Association for Computer Machinery*

ANFIS – *Adaptive Network-based Fuzzy Inference System*

BCE/UnB - Biblioteca Central da Universidade de Brasília

CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CEP – Comitê de Ética e Pesquisa

CMAC - *Cerebellar Model Articulation Controller*

cProfile – Ferramenta de *profilling* compatível com *Python*

FCE – Faculdade Ceilândia

IJCNN - *International Joint Conference on Neural Networks*

IEEE - *Institute of Electrical and Electronics Engineers*

MATPLOTLIB – Biblioteca de geração de gráficos usada com *Python*

MIMO - *Multiple Input Multiple Output*

MISO - *Multiple Input Single Output*

MLP - *Multi Layer Perceptron*

PDB – *Python Debugger*

PubMed – Site de pesquisa de artigos científicos especializado na área da saúde.

Python – Linguagem de programação orientada a objetos e funcional

LIS - Laboratório de Informática em Saúde

RNA - Redes Neurais Artificiais

tkInter – Biblioteca de interfaces gráficas usada com *Python*

TSK *fuzzy* - Takagi-Sygeno-Kang *fuzzy*

UnB - Universidade de Brasília

QTM – *Qualisys Track Manager*

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E FORMULAÇÃO DO PROBLEMA

No campo da reabilitação humana, há o subcampo que trata da construção de próteses. Estas têm como objetivo substituir a função de algum membro do corpo humano. Em especial existem próteses que substituem a função de parte das pernas, abaixo ou acima do joelho, também chamadas de próteses transtibiais para aquelas e transfemurais para estas.

As próteses também podem ser classificadas como passivas ou ativas. As passivas são compostas inerentemente por componentes mecânicos como molas e alavancas. Já as próteses ativas possuem mecanismos eletro-eletrônicos, que têm como principal função injetar energia no sistema para compensar o gasto metabólico extra, exigido por uma prótese passiva. Para as próteses ativas, faz-se necessária a criação de mecanismos de controle para as mesmas. Este mecanismo pode ser criado, usando-se engenharia de controle tradicional e/ou adaptativa, ou sistemas inteligentes, que permitem visualizar o projeto de controle para próteses (BORJIAN, 2008; SUP *et al.*, 2008).

As técnicas de engenharia de controle (GOLNARAGHI *et al.*, 2010) exigem que se criem modelos cinéticos e/ou cinemáticos para se resolver o problema em questão. Em (BORJIAN, 2008) há um exemplo de tal enfoque. Geralmente estes modelos são criados analiticamente, usando-se leis bem compreendidas da física (GARCIA, 2009). Este não é um trabalho fácil, podendo demandar muito tempo para construção e às vezes devido a grande complexidade do sistema, exige um grau maior de simplificação. Isso pode se tornar um problema sério, até mesmo inviabilizando a solução.

Sistemas inteligentes (RUSSEL *et al.*, 2010) abrangem vários tipos de tecnologias. Dentre elas, sistemas *fuzzy*, sistemas especialistas, agentes lógicos, redes neurais artificiais, etc. Esta classe de sistemas, pode facilitar bastante a gestão da complexidade, pois dependendo do caso, os modelos são bastante simples de serem construídos, pois não exigiriam toda a complexidade física e matemática da engenharia de controle.

Sistemas *fuzzy* (LILLY, 2010), por exemplo, usam como componentes de modelagem variáveis linguísticas. Exemplificando, estas em um sistema para controle da velocidade de um ventilador poderiam ser resumidas em rápida, média e lenta.

Sistemas baseados em aprendizado de máquina (*machine learning*) (BISHOP, 2006), assim como sistemas inteligentes, permitem a criação de sistemas mais simples. Isto é

possível porque estes sistemas não são programados para resolverem um problema, mas sim para "aprenderem" a resolver um problema. Por exemplo, é possível desenvolver um sistema baseado em aprendizado de máquina, que aprenderá a partir da coleta de dígitos escritos por várias pessoas, a reconhecer qualquer dígito escrito num papel por quaisquer outras pessoas.

As Redes Neurais Artificiais (RNA) (HAIKIN, 2009; BISHOP, 2006; RUSSEL *et al.*, 2010) são classificadas como sistemas baseados em aprendizado de máquina e também como sistemas inteligentes. A princípio são sistemas inspirados pela biologia do sistema nervoso central.

O uso de RNAs para controlar próteses transfermurais pode ser um atalho no processo de projeto das mesmas, pois se usando algoritmos de RNAs que já tenham sido especificados e implementados, basta alterá-los para receberem os sinais desejados e produzir as respostas esperadas.

Porém, o ciclo de marcha possui muitas variações. Por exemplo, o ciclo confortável, acelerado, subindo aclives, subindo escadas, descendo, etc. Aparentemente é possível criar RNAs para todas estas situações, mas mesmo assim, este é um trabalho hercúleo e que provavelmente exigiria muito poder computacional, o que não seria adequado para uma prótese transfemural ativa, que exige um sistema embarcado eficiente energeticamente. Com isso em mente, e inspirando-se no trabalho de (SAUBOURIN *et al.*, 2012), nota-se que seria possível modelar sistemas *fuzzy* para alterar a saída do sistema conforme a necessidade do ciclo de marchar. A vantagem de tal configuração é que um sistema *fuzzy* geralmente vai exigir menos cálculos que várias RNAs ou uma RNA com muitos milhares de ativações.

Existem vários tipos de RNA, sendo a mais popular na literatura a *Multi Layer Perceptron* (MLP). Esta é um tipo de RNA que tem sua aplicabilidade e eficiência já comprovadas em várias aplicações. A MLP, porém, tem uma desvantagem para sistemas de grande complexidade. Ela exige poder computacional considerável, pois necessita fazer muitos cálculos até definir sua saída. Uma promessa em relação a este tipo de RNA, e que inclusive já possui aplicações no mundo real, é a *Cerebellar Model Articulation Controller* (CMAC) (ALBUS, 1975). Esta é uma RNA baseada no *cerebelum* dos mamíferos. Sua vantagem em relação à MLP é que na sua forma final, ela pode ser resumida a acessos a tabelas e cálculos triviais e simples para gerar sua saída.

Neste sentido, este trabalho demonstra simulações desenvolvidas a partir de uma RNA CMAC, mostrando que com esta tecnologia é possível controlar uma prótese transfemural ativa durante o ciclo de marcha confortável. Também é objeto do trabalho, a modificação da saída da RNA CMAC, para que o sistema suporte outros ciclos de marcha através de controladores *fuzzy*.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O presente trabalho tem como objetivo principal criar um controlador para uma prótese transfemural ativa baseada em sistemas *fuzzy* e na RNA CMAC.

1.2.2 Objetivos específicos

Os objetivos específicos são:

- Definir os sinais de entrada para o controlador;
- Definir os sinais de saídas do controlador;
- Definir uma arquitetura de RNA adequada ao CMAC, a ser utilizada para controle de próteses transfemurais ativas;
- Simular um controlador CMAC com vistas ao ciclo confortável de marcha;
- Selecionar os comportamentos de exceção do ciclo de marcha confortável;
- Modelar variações do ciclo de marcha através de vários sistemas *fuzzy*;
- Criar simulações virtuais do controlador, baseados em dados reais, nos diferentes ciclos de marchas selecionados;
- Validar as simulações.

1.3 REVISÃO DA LITERATURA

Este trabalho utilizou-se de ferramentas de pesquisa bibliográfica web dos seguintes serviços: Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), *Google Scholar*, *PubMed*, *Association for Computer Machinery* (ACM), *Institute of Electrical and Electronics Engineers* (IEEE) e a Pesquisa Integrada Biblioteca Central da Universidade de Brasília (BCE / UnB).

As palavras chaves utilizadas foram: *Active transfemoral prosthesis,* CMAC, *fuzzy controller* e *artificial neural network.*

No artigo de (VILLALPANDO, 2009) é visto um estudo preliminar de uma prótese transfemural ativa, com projeto de controlador baseado em engenharia de controle. Em (SUP *et al*., 2008) é descrito um projeto completo de uma prótese transfemural ativa. O controlador da marcha nesta prótese é baseado nas funções de impedância passivas que coordenam o movimento da mesma. De fato é um controlador parecido com o do autor anterior. Em (VALLERY et al., 2011) é demonstrado um método de controle novo usando-se regressões estatísticas. Este método é baseado na coordenação das juntas da perna, por exemplo, se um joelho esta em um determinado ângulo e velocidade durante a fase de suporte, enquanto que o outro joelho deve estar em uma faixa específica de ângulo e velocidade.

Já (BORJIAN, 2008) demonstra uma prótese transfemural ativa controlada por uma base de conhecimento, que tira vantagem de uma *RNA Adaptive-Network-based Fuzzy Inference System* (ANFIS). Apesar deste controlador utilizar a ANFIS, a TNA é utilizada apenas para procurar os parâmetros desconhecidos da Takagi-Sygeno-Kang (TSK) *fuzzy*, que é o verdadeiro modelo de controlador desta prótese.

Em (LIN, 1992) é demonstrada a efetividade do modelo CMAC, através de estudos preliminares de controle cinemático e síntese de marcha. Depois de treinar uma RNA, baseada no CMAC, para aprender relações multivariáveis e não lineares, da cinemática da marcha, a RNA foi utilizada para controle de caminhada em linha reta e aclives de robôs quadrúpedes. Em (SABOURIN, 2005), demonstra-se a robustez de uma CMAC em um robô bípede em condições de marcha apresentando distúrbios. Em (SABOURIN, 2012) são apresentadas as estratégias para o uso da CMAC no mesmo tipo de robô.

Assim, com a pesquisa realizada nas bases de dados científicos, não foi encontrado na literatura nenhum projeto de prótese transfemural ativa que usasse uma RNA CMAC como controlador.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PRÓTESES TRANSFEMURAIS ATIVAS

2.1.1 Tipos de próteses

As próteses constituem uma importante contribuição da engenharia na área da reabilitação humana. Existem próteses para substituição da função de membros exteriores como braços e pernas, até a substituição de secções de esôfagos ou intestinos.

As próteses para substituir as funções das pernas, em especial, têm evoluído bastante ao longo do tempo. Existem próteses ativas e passivas para este fim. Próteses passivas são constituídas intrinsecamente por elementos mecânicos que não possuem qualquer tipo de suporte energético externo. Já próteses ativas, são constituídas por elementos mecânicos e eletrônicos, sendo que partes mecânicas da prótese recebem auxílio de energia externa para atuar na função que está sendo substituída, por exemplo, a flexão de um joelho. A principal vantagem das próteses ativas, sobre as passivas, está na compensação do débito energético que ocorre nas passivas. Esta compensação garante um ciclo de marcha mais confortável, suave e natural ao usuário da prótese ativa (BORJIAN, 2008).

2.1.2 Projeto de controle para próteses transfemurais ativas

Em (SUP *et al.*, 2008), demonstra-se o projeto de uma prótese transfemural ativas. Este trabalho escolheu técnicas de engenharia de controle para controle da prótese. Basicamente foram criados modelos matemáticos baseados na física clássica para, então, derivar uma equação de transferência que servirá de base para as leis de controle que atuarão na prótese.

Em (VALLERY *et al.*, 2011) é apresentado um enfoque novo para controle de próteses transfemurais ativas. Este modelo é baseado em regressões estatísticas, que aplicam os sinais de entrada em um modelo estatístico para então produzir um sinal de saída. Este é um modelo que pode ser considerado adaptativo.

O terceiro modelo encontrado de projeto de controle para próteses transfemurais ativas, foi o de (BORJIAN, 2008). Ele utilizou um controlador *fuzzy*, esepecificamente com o modelo Takagi-Sygeno-Kang (TSK) *fuzzy*. Este é um modelo adaptativo.

2.2 CEREBELLAR MODEL ARTICULATION CONTROLLER (CMAC)

2.2.1 Descrição

A *Cerebellar Model Articulation Controller* (CMAC) foi criada por James Sacra Albus (ALBUS, 1975). Ele se inspirou no *cerebellum* dos mamíferos para criá-la. O mesmo autor havia feito um extenso trabalho sobre o funcionamento do *cerebellum* (ALBUS 1971). Trabalho este, que resultou numa tese de Doutorado (ALBUS, 1972).

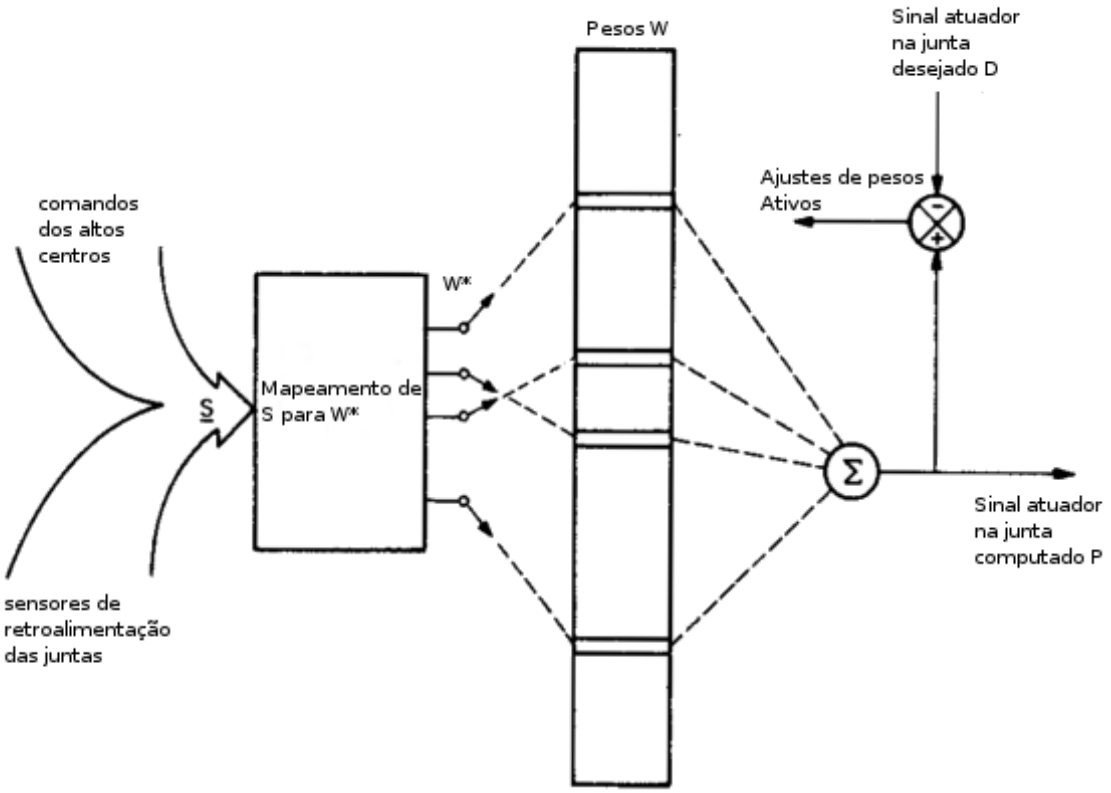

Figura 1: A CMAC para uma simples junta (ALBUS, 1975).

Na Figura 1 é possível ver o funcionamento básico da CMAC. Os sinais *S* entram no sistema, que mapeiam o mesmo para um conjunto de pesos *W** que devem ser somados para ativação. Note que apenas uma pequena fração de pesos é realmente selecionada para participar na ativação. O conjunto de pesos disponíveis na CMAC é necessariamente maior que o número de pesos ativados *W**.

A CMAC da Figura 1 também pode ser classificada como um sistema *Multiple Input Single Output* (MISO), ou seja, suporta a entrada de vários sinais de entrada e processa um sinal de saída. Para se produzir uma CMAC *Multiple Input Multiple Output* MISO, bastaria implementar várias MISOs, compartilhando as mesmas entradas.

Sendo a CMAC uma RNA com aprendizado supervisionado, seus pesos podem ser atualizados simplesmente computando-se o erro e atualizando-se apenas os pesos que participaram do computo do sinal *P*. Isto é tudo para o treinamento da CMAC.

3 METODOLOGIA

3.1.1 Coleta dos Dados

Os dados para o treinamento da RNA CMAC são dados cinemáticos, capturados através de *motion capture*, utilizando-se de várias câmeras *Qualisys Oqus* MRI, com marcadores passivos e pacote de software QTM 3.2 da *Qualisys*. O sistema utilizado suporta até 74 canais, ou marcadores simultâneos.

O projeto no qual ocorreu a coleta foi aprovado pelo Comitê de Ética da Faculdade de Saúde da UnB, processo N11911/12 (ver Anexo 1.)

A Figura 2 mostra o processo para coleta de dados.

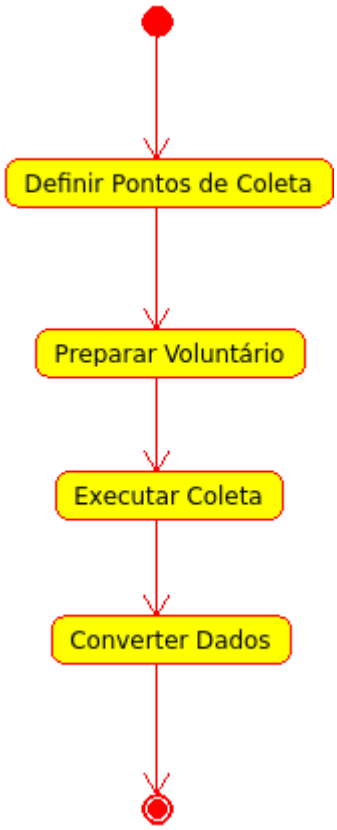

Figura 2: Processo de coleta de dados.

Primeiro deve-se definir o voluntário da coleta e determinar o dia para este processo. Além disso, também é necessário definir quais os pontos no corpo do voluntário devem ser mapeados. Também se devem distribuir os marcadores em várias posições ao

longo das pernas. Como só a flexão e a extensão dos joelhos interessam para este trabalho, utilizam-se somente marcadores nas tíbias, joelhos e trocânteres das duas pernas.

O próximo passo se refere ao voluntário, isto é, ele deve repetir um ciclo de marcha confortável de aproximadamente 5 segundos, por 5 vezes na frente das câmeras.

Quanto aos dados, estes devem ser convertidos para formato adequado à linguagem *Octave*, que é a mesma opção para converter para o *MATLAB*. Esta opção é própria do *QTM*. Além da conversão é necessário definir o nome de cada item na matriz de dados coletados. Cada coluna desta matriz representa um marcador, são estes pontos que devem ser nomeados. Por exemplo, coluna 1 igual ao trocânter direito. O número que o *QTM* atribui internamente ao marcador é a posição do marcador na matriz. Este número é chamado dentro do *QTM* de canal. Os dados trazem variáveis espaciais e o erro, com respeito à posição (X, Y, Z) dos marcadores.

A disposição que os dados obtidos neste processo se apresentam, é conforme se encontra na Figura 3.

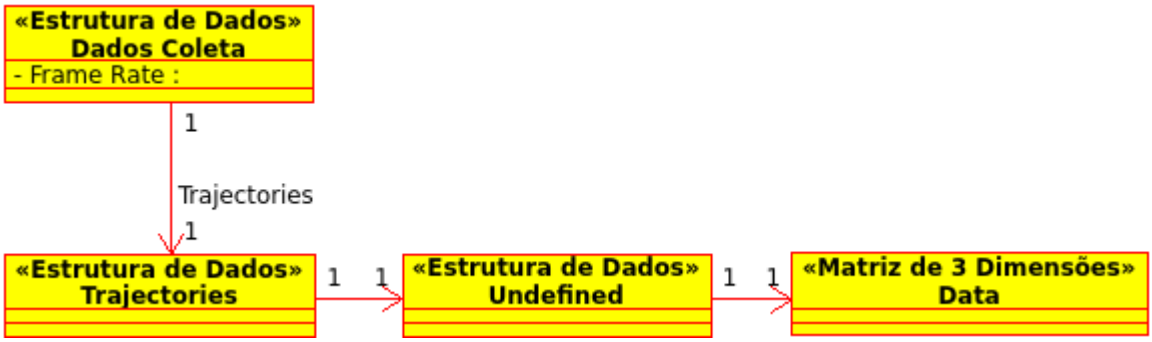

Figura 3: Dados disponibilizados pelo *QTM*.

Os únicos que interessam são o *Frame Rate* e o *Data*.

São retornados vários dados, mas os de interesse para o projeto são os que estão na Figura 3. O *Frame Rate* é a taxa de coleta dos dados e está em segundos. A matriz de 3 dimensões está disposta da seguinte forma:

1. A primeira dimensão é 74 e representa o número de canais do sistema de coleta;

2. A segunda dimensão é 4 e representa a posição num plano 3D (X, Y, Z) do marcador, mais o erro;

3. A terceira dimensão é número de *frames* coletados numa caminhada específica. Este número é variável.

3.1.2 Extração e transformação dos dados

Com os dados necessários disponibilizados no formato adequado é possível fazer os cálculos de angulações, velocidades angulares e acelerações angulares dos joelhos.

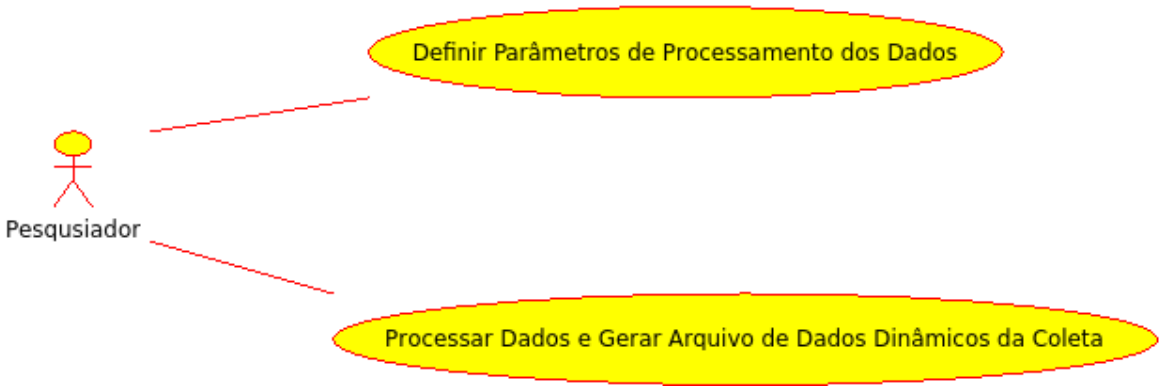

Figura 4: Casos de Uso para Extração e .transformação dos dados.

O caso de uso Definir Parâmetros de Processamento dos Dados, definido na Figura 4, consiste em se definir os dados necessários para que depois seja possível processar os dados coletados. Estes dados são os definidos na Figura 5Figura 1. O valor dos 6 primeiros atributos da estrutura de dados *Configuração do Processamento,* são os canais usados no QTM para tais marcadores.

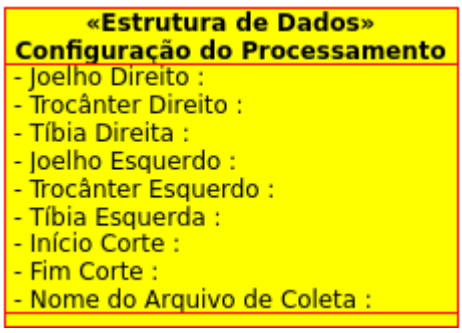

Figura 5: Dados para processamento da coleta.

O caso de uso Processar Dados e Gerar Arquivos de Dados Cinemáticos, definido na Figura 4, da coleta deve obedecer o processo da Figura 6.

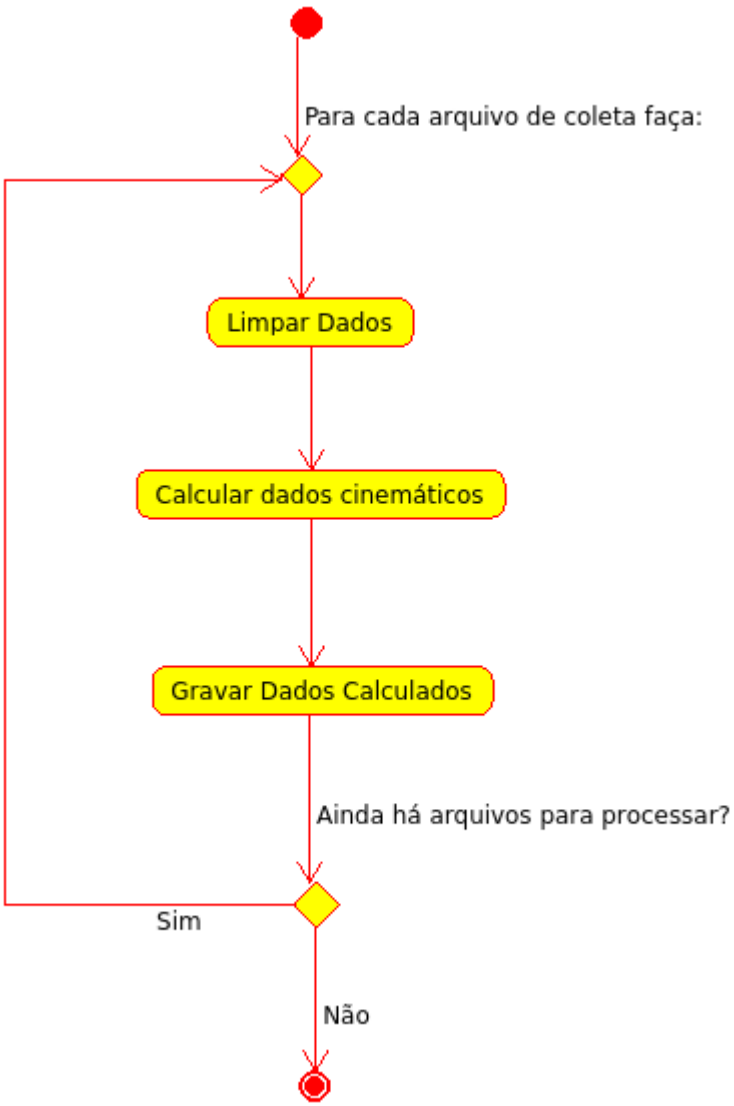

Figura 6: Processo de tratamento dos dados.

A limpeza dos dados consiste em retirar os dados desnecessários, como marcadores não desejados e retirada de *frames* do início e/ou final de um arquivo de coleta, que não estejam no ciclo de marcha confortável.

Os cálculos realizados devem ser as velocidades instantâneas, velocidades angulares e acelerações angulares.

Velocidades instantâneas dos joelhos são calculadas conforme a Equação 1.

$$\vec{v} = (\vec{a} - \vec{b})/t \quad (1)$$

A variável $\vec{a}$ é a posição (X, Y, Z) do joelho em uma determinada leitura sequencial dos dados coletados. A variável $\vec{b}$ é a próxima posição (X, Y, Z) da sequência.

Para o cálculo das angulações dos joelhos, primeiro os marcadores destes devem ser transladados para uma origem, assim é possível se usar a Equação 5 como descrita em (EDWARDS, 2006). Para tal, usam-se as Equações 2, 3 e 4.

$$\vec{u} = \vec{t} - \vec{j} \quad (2)$$

$$0 = \vec{j} - \vec{j} \quad (3)$$

$$\vec{x} = \vec{b} - \vec{j} \quad (4)$$

A variável $\vec{t}$ é o vetor referente ao trocânter de uma perna. A variável $\vec{j}$ é a variável referente ao joelho. A variável $\vec{b}$ é a variável referente à tíbia. A variável $\vec{u}$ é o trocânter transladado para uma origem. A posição do joelho é sempre 0, pois o joelho é a origem que servirá de base para o cálculo do ângulo. A variável $\vec{v}$ é a posição da tíbia transladada. A translação de vetores é documentada em (POOLE, 2011).

Agora que se tem os pontos transladados para uma origem, pode-se usar a Equação 5 para o cálculo do ângulo do joelho.

$$\theta = \cos^{-1}(\vec{u}.\vec{v}/\|\vec{u}\|.\|\vec{v}\|) \quad (5)$$

O operador $\| \ \|$ é o cálculo da distância euclidiana, veja equação 3, ou norma. Pode ser calculado, segundo (POOLE, 2011), de acordo com a Equação 6. Resumindo, é a raiz quadrada do produto interno de um vetor.

$$\|\vec{u}\| = \sqrt{\vec{u} \cdot \vec{u}} \quad (6)$$

As velocidades angulares dos joelhos são calculadas a partir da equção 7.

$$\omega = (\theta_1 - \theta_2)/t \quad (7)$$

A variável θ_1 é o ângulo de um joelho num determinado *frame*. A variável θ_2 é exatamente o ângulo do próximo *frame*. A variável t é o *frame rate*, oriundo dos dados da coleta.

A última etapa deste processo é a gravação dos dados para que possam ser usados pela RNA CMAC. Estes dados devem ser gravados num arquivo em formato texto. As linhas neste arquivo equivalem aos *frames*. Cada coluna equivale às informações, na ordem em que aparecem, da estrutura de dados descrita na Figura 7.

«Estrutura de Dados» Dados Cinemáticos
Velocidade Angular do Joelho Esquerdo
Velocidade Angular do Joelho Direito
Ângulo do Joelho Esquerdo
Ângulo do Joelho Direito
Aceleração Angular do Joelho Esquerdo
Aceleração Angular do Joelho Direito
Posição no plano X do Joelho Esquerdo
Posição no plano Y do Joelho Esquerdo
Posição no plano Z do Joelho Esquerdo
Posição no plano X do Joelho Direito
Posição no plano Y do Joelho Direito
Posição no plano Z do Joelho Direito

Figura 7: Dados Cinimáticos da Marcha.

3.1.3 Construção de uma RNA CMAC

Modelo geral

A RNA CMAC proposta para este trabalho é resumida na Figura 8.

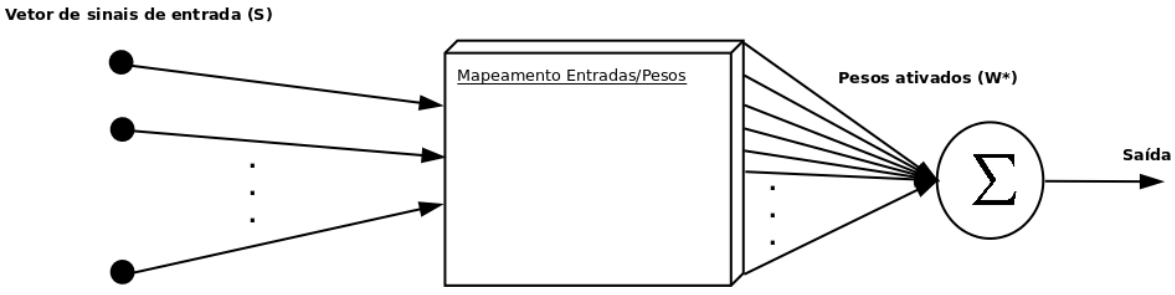

Figura 8: CMAC resumida.

A variável *S* é o vetor de sinais de entrada. Esses sinais são passados para um processo de mapeamento entre a entrada e um conjunto de pesos. Depois apenas os pesos ativados participam da somatória que é o sinal de saída.

Tendo este modelo em mente são necessários os passos descritos a seguir (as explicações a seguir foram feitas baseadas em (ALBUS, 1975)).

Primeiramente, define-se o número de pesos *NW** a serem ativados para comporem a saída da CMAC.

O segundo passo é definir os possíveis valores para cada item do vetor de entradas *S*. Por exemplo, se o primeiro item *s1* de *S* aceita valores de -1 até 1 e se quer 5 valores possíveis, discretiza-se então os valores de -1 até 1, conforme Figura 9.

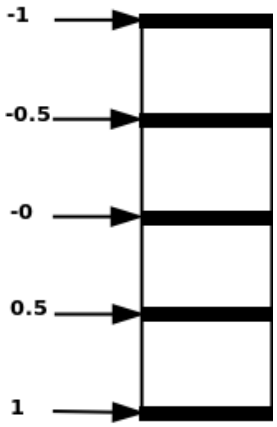

Figura 9: Discretização de *s1*.

Isto significa que quaisquer que sejam os valores de *s1* os mesmos devem ser convertidos para -1, -0,5, 0, 0,5 e 1. Por exemplo, se o valor de *s1* for 0,75, será convertido para o valor 1, se for -0,75 será o valor 0 e se for 0,25 será o valor 0,5. Portanto, a discretização dá-se o nome de resolução da CMAC.

O próximo passo é criar uma tabela para cada um dos sinais discretizados de entrada do vetor *S*. Supondo que o vetor *S* possui 2 sinais de entrada *s1* e *s2 e* um número de ativações *NW** igual a 3, cria-se Tabela 1 e a Tabela 2. Para facilitar o entendimento, irá se considerar os valores de *s1* iguais aos inteiros de 1 até 6 e os valores de *s2* iguais aos inteiros de 1 até 4.

Tabela 1: Mapeamento de *s1*

Valores de *s1*	Mapeamento *m1*
1	0, 1, 2
2	3, 1, 2
3	3, 4, 2
4	3, 4, 5
5	6, 4, 5
6	6, 7, 5

Tabela 2: Mapeamento de *s2*

Valores de *s2*	Mapeamento *m2*
1	0, 1, 2
2	3, 1, 2
3	3, 4, 2
4	3, 4, 5

Estas tabelas são criadas da seguinte forma: O mapeamento consiste num número de itens igual a *NW**, 3 no caso. Para a primeira linha de cada uma das tabelas, atribui-se uma sequência de 3 valores inteiros começando com 0. A próxima linha deve conter o próximo valor da sequência, 3, como primeiro item do mapeamento e continuar com os demais itens iguais aos da linha anterior. Na próxima linha, substitui-se o segundo item de

mapeamento pelo próximo número da sequência, 4, mantendo-se os demais itens e assim sucessivamente.

Depois de mapeado cada valor de cada item de entrada, deve-se combinar os mapeamentos de acordo com a Tabela 3.

Tabela 3: Mapeamento para os pesos *W.*

s2 / *s1*	**1**	**2**	**3**	**4**
1	(0, 0), (1, 1), (2, 2)	(0, 3), (1, 1), (2, 2)	(0, 3), (1, 4), (2, 2)	(0, 3), (1, 4), (2, 5)
2	(3, 0), (1, 1), (2, 2)	(3, 3), (1, 1), (2, 2)	(3, 3), (1, 4), (2, 2)	(3, 3), (1, 4), (2, 5)
3	(3, 0), (4, 1), (2, 2)	(3, 3), (4, 1), (2, 2)	(3, 3), (4, 4), (2, 2)	(3, 3), (4, 4), (2, 5)
4	(3, 0), (4, 1), (5, 2)	(3, 3), (4, 1), (5, 2)	(3, 3), (4, 4), (5, 2)	(3, 3), (4, 4), (5, 5)
5	(6, 0), (4, 1), (5, 2)	(6, 3), (4, 1), (5, 2)	(6, 3), (4, 4), (5, 2)	(6, 3), (4, 4), (5, 5)
6	(6, 0), (7, 1), (5, 2)	(6, 3), (7, 1), (5, 2)	(6, 3), (7, 4), (5, 2)	(6, 3), (7, 4), (5, 5)

O mapeamento da Tabela 3 é feito da seguinte forma: Coloca-se cada valor discretizado de *s1* nomeando cada uma das linhas da tabela. Coloca-se cada valor discretizado de *s2* nomeando cada uma das colunas das tabelas, sem contar a primeira coluna que são os valores de *s1*. Cada célula da tabela é obtida recuperando-se os itens de *s1* e de *s2*, que estão na Tabela 1 e na Tabela 2, e concatenando-se cada item de acordo com sua posição. Por exemplo, para o valor de *s1* igual a 5, recuperam-se os itens (6, 4, 5), para o valor de *s2* igual a 2, recuperam-se os itens (3, 1, 2). Agora é só criar novos itens, na mesma quantidade do valor *NW**, concatenando-se cada item de acordo com sua posição ((6, 3), (4, 1), (5, 2)). Cada um destes três novos itens é uma chave para se acessar uma tabela de pesos *W*.

Para se calcular a saída do sistema, suponha que os valores de *s1* e *s2*, ainda sejam 5 e 2, respectivamente, e *NW** ainda seja 3. Neste caso, recupera-se o item da linha 5 e da

coluna 2, que é ((6, 3), (4, 1), (5, 2)). Agora é acessada a tabela de pesos *W* e são recuperados os pesos cujas chaves são (6, 3), (4, 1) e (5, 2). Note-se que serão recuperados exatamente 3 pesos, que é exatamente o número de ativações *NW** desejadas. Os valores recuperados constituem um vetor chamado *W**. A saída é calculada somando-se os valores de *W**.

Para uma CMAC com mais de 2 sinais no vetor *S*, deve-se proceder como descrito acima para os dois primeiros sinais. Deve-se fazer o mapeamento inicial do próximo sinal, igual ao descrito acima, ver Tabela 4. Depois disto cria-se uma nova tabela cuja primeira coluna é composta de cada uma das colunas da Tabela 3. As demais colunas da nova tabela são nomeadas com os valores discretizados do próximo sinal. Concatena-se então o valor de cada item da primeira coluna, com cada item de mapeamento da Tabela 4.

Tabela 4: Mapeamento de s3

Valores de *s3*	**Mapeamento *m3***
1	0, 1, 2
2	3, 1, 2

Supondo-se que o próximo sinal *s3* possui os valores 1 e 2, deve ser criada então a

Tabela 5.

Tabela 5: Tabela de mapeamento final para 3 sinais de entrada (resumida). A primeira linha são os valores de *s3*.

	1	**2**
Células da Tabela 4		
(0, 0), (1, 1), (2, 2)	(0, 0, 0), (1, 1, 1), (2, 2, 2)	(0, 0, 3), (1, 1, 1), (2, 2, 2)
(3, 0), (1, 1), (2, 2)	(3, 0, 0), (1, 1, 1), (2, 2, 2)	(3, 0, 3), (1, 1, 1), (2, 2, 2)
...	...	...
(6, 0), (7, 1), (5, 2)	(6, 0, 0), (7, 1, 1), (5, 2, 2)	(6, 0, 3), (7, 1, 1), (5, 2, 2)
(3, 3), (1, 1), (2, 2)	(3, 3, 0), (1, 1, 1), (2, 2, 2)	(3, 3, 3), (1, 1, 1), (2, 2, 2)
(3, 3), (4, 1), (2, 2)	(3, 3, 0), (4, 1, 1), (2, 2, 2)	(3, 3, 3), (4, 1, 1), (2, 2, 2)
...	...	...
(6, 3), (7, 4), (5, 5)	(6, 3, 0), (7, 4, 1), (5, 5, 2)	(6, 3, 3), (7, 4, 1), (5, 5, 2)

Para o próximo sinal, cria-se uma nova tabela contendo na primeira coluna, todas as colunas da

Tabela 5, concatenando-se, então, o valor de cada item da primeira coluna com cada item de mapeamento do sinal em questão e assim sucessivamente a primeira linha possui itens (0,0), (1, 1) e (2, 2), os itens de s3 para o seu valor 1 da primeira coluna são (0, 1, 2), o resultado da concatenação é (0, 0, 0), (1, 1, 1), (2, 2, 2). Por exemplo, nota-se, ainda, que o número de pesos total é igual ao número de itens diferentes, encontrados em cada célula da tabela de mapeamento final.

Depois de gerada a tabela final, que incorpora o último sinal do vetor de entradas, cria-se um dicionário, para cada item em cada célula desta tabela, isto é, cria-se uma entrada no dicionário. Na tabela haverá itens duplicados, mas no dicionário não. Por exemplo, na

Tabela 5, a célula da primeira linha e primeira coluna.

3.1.4 Treinamento da CMAC

Para se treinar a CMAC, primeiro define-se o conjunto de dados para treinamento que devem ser usados. Pode ser usado algo entre 20% e 50%, dos dados coletados para desenvolvimento do sistema. Cada item destes dados deve conter um vetor de sinais de entradas *S* e a resposta desejada *D* para estes sinais. A tabela de pesos deve ser inicializada com valores randômicos. Para cada item deve ser calculado o vetor *W**, que contém os endereços dos pesos a serem ativados. Calcula-se então a saída da rede *P* para o vetor *S* conforme o item 3.1.3. Atualizam-se os pesos sinápticos conforme a Equação 6.

$$w_i = w_i + [\alpha(D - P)] / NW* \tag{6}$$

A variável w_i é um item qualquer dentro da tabela de pesos *W*. Em cada iteração no conjunto de dados para treinamento, devem-se atualizar apenas os pesos descritos em *W** naquele momento. A variável α é o coeficiente de aprendizado, deve ser um valor entre 0 e 1. A variável *NW** é o número de valores a serem utilizados para ativação.

Deve-se determinar o número de iterações *batch* para o sistema. Uma iteração *batch* consiste no processamento dos pesos para cada item dentro do conjunto de dados para treinamento. O sistema deve continuar iterando até atingir o número de iterações *batch*.

Apesar do número de iterações *batch* ser válido como critério de parada para o treinamento da CMAC, nada impede o uso de outros critérios, por exemplo, o erro quadrado médio da saída *Y* em relação ao desejado *D*.

3.1.5 Otimização da CMAC

São apenas 4 parâmetros, que realmente precisam ser otimizados para uma CMAC, como a descrita na seção 2.2.

- O número de valores possíveis para cada sinal de entradas;
- O número de ativações de pesos que participam no cálculo da saída;
- O número de iterações *batch*;

- A taxa de aprendizado.

O seguinte processo deve ser executado para se otimizar os parâmetros:

1. O número de valores possíveis para cada sinal de entrada deve variar de 10 em 10 entre os valores 10 e 200;
2. O número de ativações dos pesos deve variar 1 em 1 entre os valores 1 e 30;
3. O número de iterações *batch* pode ser 50, 100 ou 150.
4. A taxa de aprendizado deve varia de 0.1 em 0.1 de 0 até 1.
5. Deve-se iterar para cada combinação de valores descritos entre 1 e 4, calculando-se o erro quadrado médio acumulado para cada combinação possível. O menor erro calculado representa o conjunto de parâmetros desejados.

Este esquema não visa achar valores de parâmetros ótimos, mas tão somente valores que tornem a solução do problema em questão viável.

3.2 PROPOSTA INICIAL

3.2.1 Uso da CMAC

Para vetores de entradas semelhantes, conjuntos de pesos semelhantes são ativados. Esta característica é que garante a CMAC seu poder de generalização. Quando vetores de entradas muito diferentes são apresentados à CMAC o conjunto de pesos ativados tende a ser bastante dissemelhante. Estas características conferem a CMAC sua aplicabilidade no controle de próteses transfemurais ativas, pois, por exemplo, um joelho não muda sua velocidade angular em valores muito grandes instantaneamente, ele vai acelerando de uma velocidade a outra de uma forma biomecanicamente plausível, alterando-se de tempos em tempos. De acordo com a teoria da CMAC, isto quer dizer que num ciclo de marcha confortável, a cada nova entrada, pelo menos alguns pesos da entrada anterior serão ativados.

3.3 DELIMITAÇÃO DO ESTUDO

O presente trabalho visa o controle de próteses tranfemurais ativas. De modo algum busca a construção de próteses em si. A ideia é criar pacotes de software suficientes para se projetar o controle de tais próteses. Por exemplo, estes pacotes devem ser capazes de extrair os dados provenientes da *motion capture*, transformá-los em informações de velocidades instantâneas de pontos do corpo, velocidades angulares de joelhos e tornozelos e acelerações angulares de joelhos e tornozelos. Também deve ser capaz de auxiliar no projeto de CMACs embarcadas nas próteses, permitindo a seleção de combinações de sinais de entradas diferentes e aproximando a saída de sinal desejada.

Também faz parte do estudo o projeto de um controlador para uma prótese transfemural ativa. Este controlador deve aproximar um sinal de saída, provavelmente a velocidade angular de um joelho. Como sinal de entrada, deve-se estudar várias combinações de entradas diferentes, por exemplo, as velocidades instantâneas num plano em 3D (X, Y, Z) dos joelhos mais a velocidade angular do joelho contrário ao joelho que se está aproximando a velocidade angular.

Também deve ser analisado o uso de controladores *fuzzy* para se alterar para ciclos de marcha que não sejam confortáveis, por exemplo, subir escadas, ou subir aclives, descer declives, marcha rápida, etc. O que busca é mostrar a viabilidade de tal esquema.

3.4 RESULTADOS ESPERADOS

Espera-se que ao final do projeto atinjam-se os seguintes resultados:

- Pacote de software para extração e transformação de dados coletados por *motion capture;*
- Pacote de software para manipulação de RNAs CMACs com as seguintes funcionalidades:
    - Treinamento de RNAs CMACs;
    - Auxílio no projeto de controladores CMACs;
    - Otimização de parâmetros de uma CMAC;

- Programação de controladores *fuzzy* para atender ciclos de marcha diferentes do ciclo confortável de marcha;

- Controlador CMAC que aproxime a velocidade angular de um joelho no ciclo confortável de marcha.

3.5 RESULTADOS PARCIAIS

3.5.1 Pacote de software Gait Data Loader

Este pacote já está funcional e é capaz de extrair os dados das posições, num plano 3D (X, Y, Z), das posições dos trocânteres, joelhos e tíbias da coleta de dados feita por *motion capture*. O software também calcula velocidades instantâneas, angulações, velocidades angulares e acelerações angulares, todas dos joelhos. Ou seja, todo o item 3.1.2 da metodologia foi implementado em *Octave* e disponibilizado no site https://github.com/rob-nn/gait_data_loader. Usando-se a ferramenta *Git* é possível baixar todo material de uma vez só.

3.5.2 Pacote de software Motus

Este pacote já possui ferramentas para treinamento de CMACs e auxílio no projeto de controladores de próteses transfemurais ativas.

Foram implementadas as classes de objetos em *Python* mostradas na Figura 10.

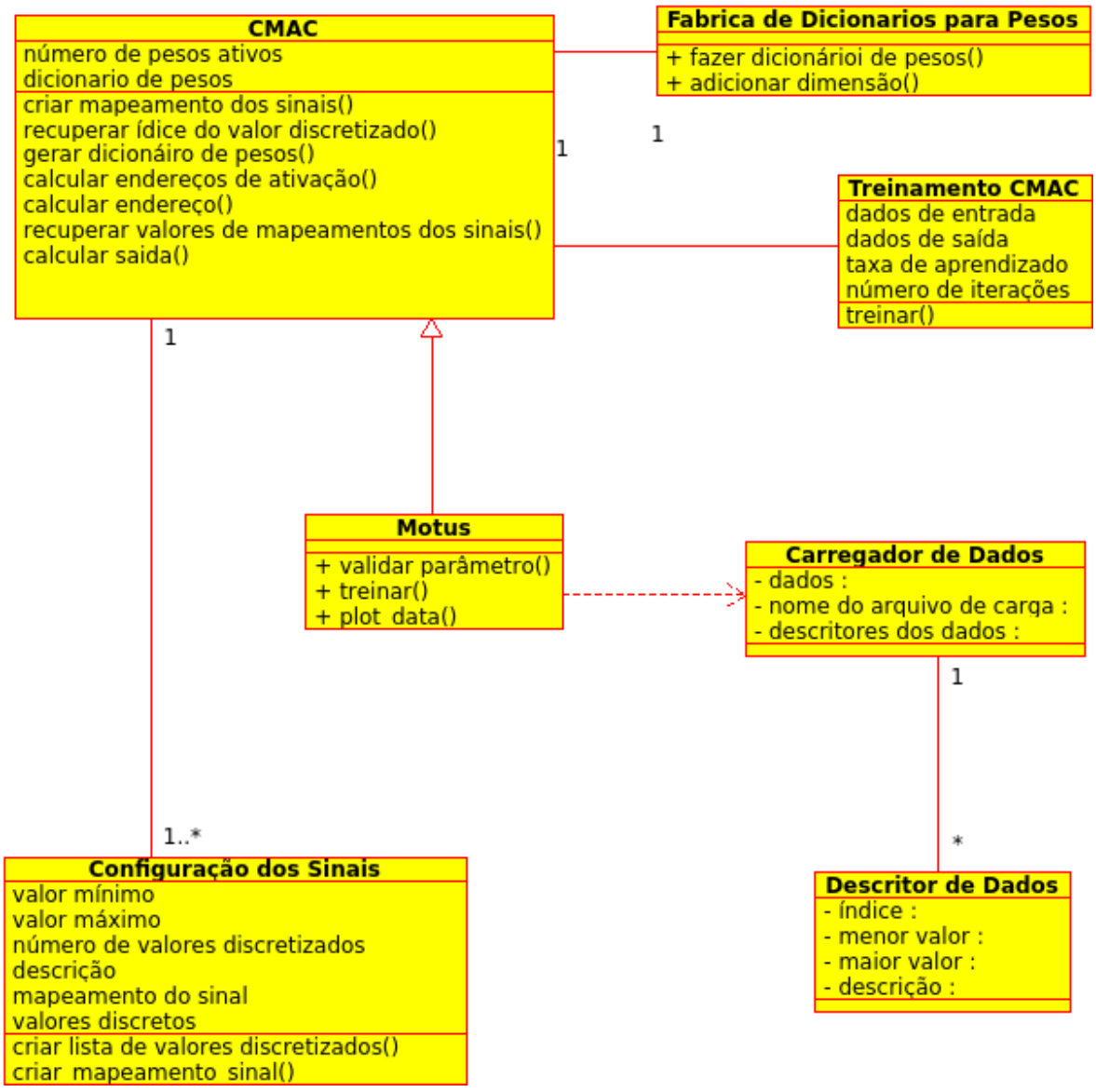

Figura 10: Classes implementadas.

O código fonte deste software está em https://github.com/rob-nn/motus.

O software pode ser executado via linha de comando digitando-se: python motus.py.

Este software já está configurado com três variáveis de entrada, posições no espaço X, Y, Z, do joelho esquerdo. Como saída ele aproxima a velocidade angular do joelho direito. Ele também usa como dados de entrada o arquivo de marcha da terceira coleta. São discretizados 15 valores para cada uma das posições e executadas 50 iterações. Lembrando que cada iteração consiste do processamento de todos os dados do arquivo de coleta.

O resultado da execução do programa pode ser visto na Figura 11.

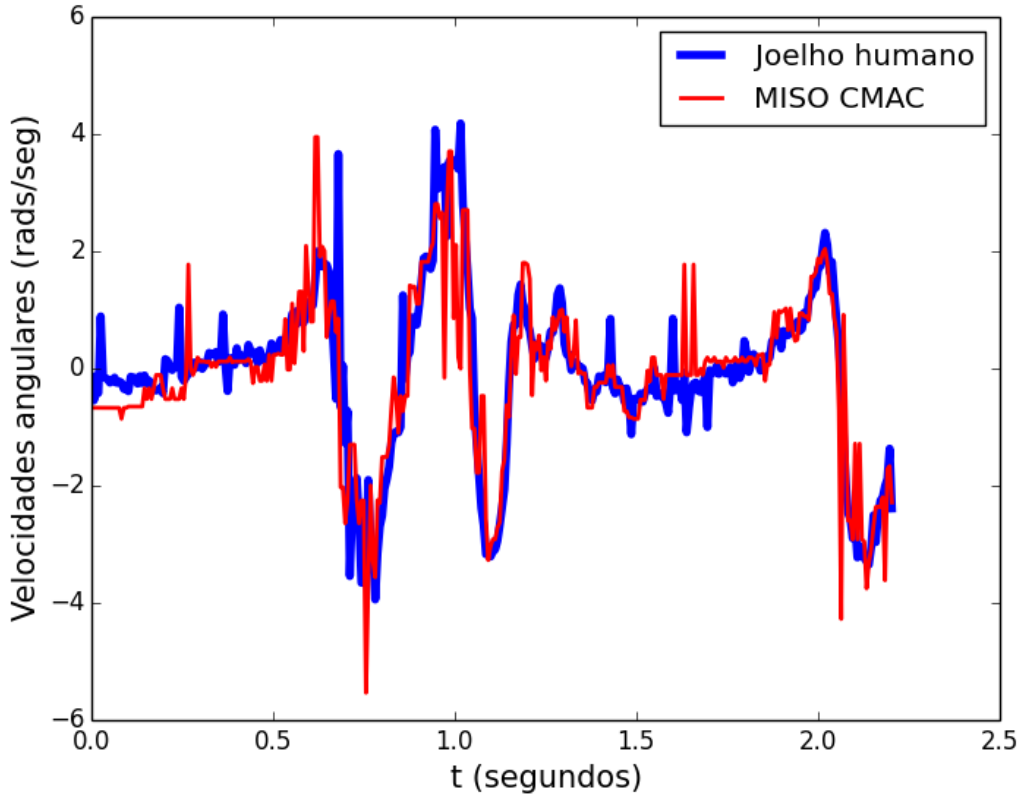

Figura 11: Resultado da execução da CMAC.

Na Figura 11 verifica-se a linha vermelha representando o resultado da CMAC. A linha azul são os dados coletados para o joelho direito do humano. Foram usados 30% dos dados da terceira coleta para treinamento escolhidos aleatoriamente. Os demais 70% foram passados para CMAC resultando na Figura 11.

Além disso, foi construído o módulo motusgui.py. Este módulo permite fazer visualmente a configuração de várias variáveis diferentes, além de configurar o sinal de saída, o número de ativações e o número de iterações. Ver Figura 12.

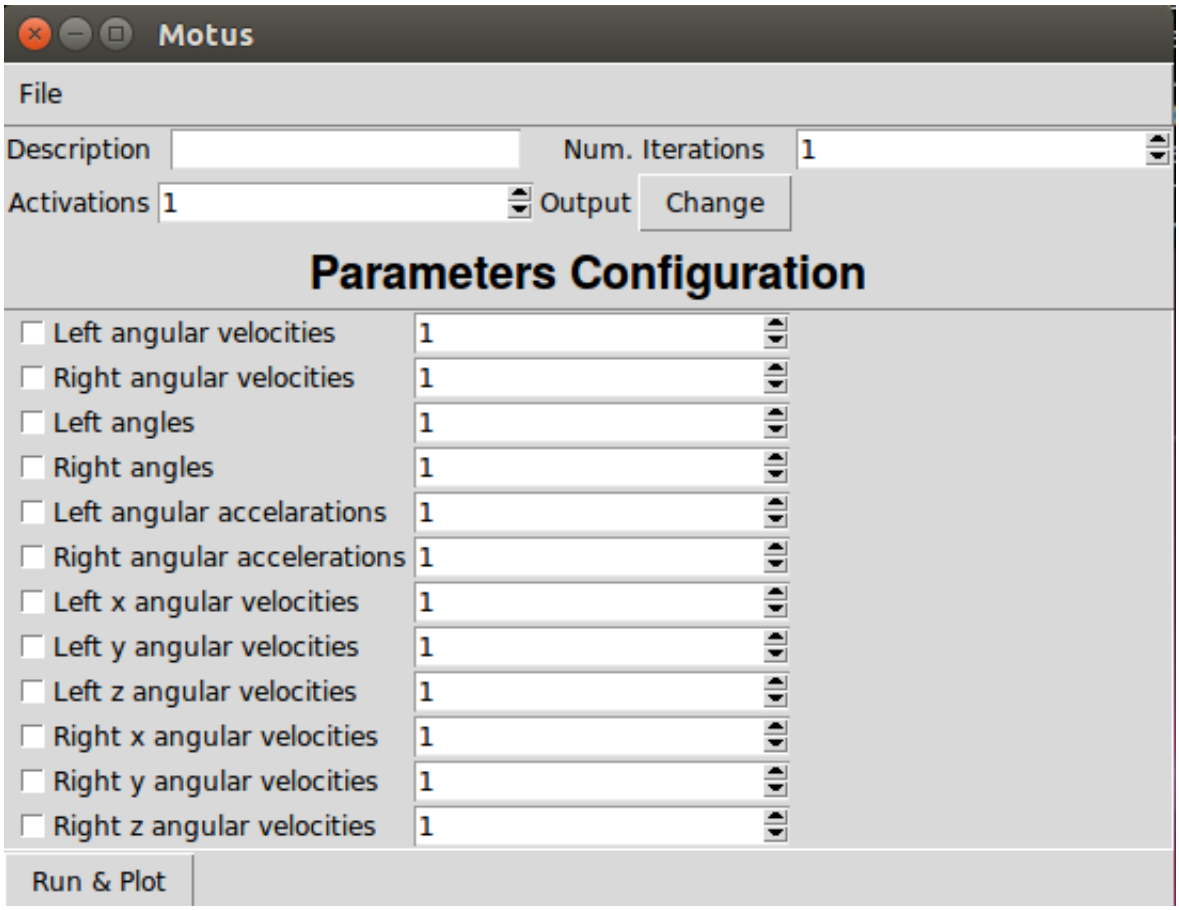

Figura 12: Motus GUI.

O resultado de se cilcar no botão Run &Plot da Figura 12, é o mesmo do programa motus.py, ou seja um gráfico como o da Figura 11.

4 VIABILIDADE DA PESQUSA

4.1 COLETA E ANÁLISE DOS DADOS

Até o momento, como descrito anteriormente, já foram coletados dados suficientes para se aproximar sinais de saída de modo satisfatório. O próximo passo é coletar dados para ciclos alternativos ao ciclo confortável de marchar e criarem-se controladores *fuzzy* que modifiquem a saída do controlador CMAC para adequar os sinais de saída a eles. Para tal atividade ocorrer, deve-se realizar a captura de tais dados no início do semestre que vem (1/2015).

4.2 CRONOGRAMA

A Tabela 6 indica as etapas para se chegar a defesa de dissertação.

Tabela 6: Cronograma.

ATIVIDADES	2015					
	JAN	FEV	MAR	ABR	MAI	JUN
Artigo *Word Congress in Medical Physic and Biomedical Engineering* (2015)		X				
Artigo *International Joint Conference on Neural Networks* (IJCNN) 2015	X					
Artigo de Revista IEEE *Transactions on Biomedical Engineering*			X			
Terminar desenvolvimento do Data Loader	X					
Terminar desenvolvimento do Motus	X	X				
Desenvolver controladores *fuzzy*			X	X		
Coletar dados de ciclo alternativo			X			
Escrever a dissertação	X	X	X	X		
Revisar e entregar a dissertação					X	
Defesa Final						X

4.3 RECURSOS TECNOLÓGICOS

A programação do sistema é realizada em linguagem *Python* 2.7, usando sistemas operacionais Mac OS X 10.8 ou Ubuntu 14.07. Foi usado o Python DeBuger (PDB) para *debuging*, a biblioteca *Unittest* para testes unitários e *cProfile* para *profilling*. Todas *open source* e pertencentes ao próprio *Python*, que também é *open source*. Para desenvolvimento de interfaces gráficas foram usadas as bibliotecas *Matplotlib* 1.4.2 para gráficos e o *Tkinter*, do próprio *Python*, para telas em geral. Ambas são *open source*. Como editor de texto utilizou-se o *Vim* 7.3.

Todo o código desenvolvido foi disponibilizado no *Github* no endereço: https://github.com/rob-nn/motus.

Além do software de implementação da CMAC, também foi desenvolvido um software de extração de dados escrito em *Octave* 3.8.1, ferramenta esta *open source*, que também está no *github* no endereço: http://github.com/rob-nn/gait_data_loader.

Também foram usadas máquinas dedicadas ao processamento, isto é, computadores MAC-Pró disponibilizados pelo Laboratório de Informática em Saúde (LIS).

4.4 RESTRIÇÕES

Novas coletas de dados devem ser marcadas com antecedência, devido ao uso do laboratório.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBUS, J. S. Theoretical and experimental aspects of a cerebellar model. [s.l.] University of Mariland, 1972.

ALBUS, S. A Theory of Cerebellar Function. v. 10, p. 25–61, 1971.

BISHOP, C. M. Pattern Recognition and Machine Learning. Singapore: Springer, 2006. v. 4p. 738

BORJIAN, R. Design , Modeling , and Control of an Active Prosthetic Knee. [s.l: s.n.]. p. 17–32

GARCIA, C. Modelagem e Simulação de Sistemas Industriais. 2. ed. São Paulo: edUSP, 2009.

GOLNARAGHI, F.; KUO, C. B. Automatic Control Systems. 9. ed. a: John Wiley & Sons, Inc., 2010.

HAYKIN, S. Neural Networks and Learning Machines. 2. ed. New Jersey: [s.n.]. p. New Jersey

LILLY, J. H. Fuzzy Control and Identification. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2010.

PERRY, J. Gait Analysis Normal and Pathological Function. Thorofare: SLACK Inc., 1992.

RUSSEL, J. S.; NORVIG, P. Artificial intelligence a Modern Approach. 3. ed. New Jersey: Pearson, 2010.

SABOURIN, C.; YU, W.; MADANI, K. Gait Pattern Based on CMAC Neural Network for Robotic Applications. Neural Processing Letters, v. 38, n. 2, p. 261–279, 20 nov. 2012.

SUP, F.; BOHARA, A.; GOLDFARB, M. Design and Control of a Powered Transfemoral Prosthesis. The International journal of robotics research, v. 27, n. 2, p. 263–273, 1 fev. 2008.

ANEXO

1. PROCESSO NO COMITÊ DE ÉTICA

Universidade de Brasília
Faculdade de Ciências da Saúde
Comitê de Ética em Pesquisa – CEP/FS

PROCESSO DE ANÁLISE DE PROJETO DE PESQUISA

Registro do Projeto no CEP: **119/11**
Título do Projeto: "Tecnologias avançadas de próteses para amputados de membro inferior".
Pesquisadora Responsável: Geovany Araujo Borges
Data de Entrada: 31/08/11

Com base na Resolução 196/96, do CNS/MS, que regulamenta a ética em pesquisa com seres humanos, o Comitê de Ética em Pesquisa com Seres Humanos da Faculdade de Ciências da Saúde da Universidade de Brasília, após análise dos aspectos éticos e do contexto técnico-científico, resolveu **APROVAR** o projeto **119/11** com o título: "Tecnologias avançadas de próteses para amputados de membro inferior". Área Temática Especial – "Pesquisa Grupo I Novos Procedimentos, Novos Equipamentos" analisado na 3ª reunião ordinária realizada no dia 12 de março de 2013.

O pesquisador responsável fica, desde já, notificado da obrigatoriedade da apresentação de um relatório semestral e relatório final sucinto e objetivo sobre o desenvolvimento do Projeto, no prazo de 1 (um) ano a contar da presente data (item VII.13 da Resolução 196/96).

Brasília, 14 de março de 2013.

Natan Monteiro de Sá
coordenador do CEP-FS/UnB

Comitê de Ética em Pesquisa com Seres Humanos - Faculdade de Ciências da Saúde
Universidade de Brasília - Campus Universitário Darcy Ribeiro - CEP: 70.910-900
Telefone: (61)-3107-1947 Email: cepfs@unb.br