Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Refatoração do Sistema NeuroR

Dasayeve Kaique Souza de Oliveira Xavier

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Gilda Aparecida de Assis

Dezembro, 2019
João Monlevade–MG

Dasayeve Kaique Souza de Oliveira Xavier

Refatoração do Sistema NeuroR

Orientador: Gilda Aparecida de Assis

Monografia apresentada ao curso de Engenharia da Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Dezembro de 2019

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Dasayeve Kaique Souza de Oliveira Xavier

Refatoração do Sistema NeuroR

Membros da banca

Harlei Miguel de Arruda Leite- Doutor- UFOP
Luiz Carlos Bambirra Torres- Doutor- UFOP

Versão final
Aprovado em 16 de Dezembro de 2019

De acordo

Gilda Aparecida de Assis

Documento assinado eletronicamente por **Gilda Aparecida de Assis**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/01/2020, às 16:08, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0030355** e o código CRC **0C320BBA**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Este trabalho é dedicado a meus pais.

Agradecimentos

Agradeço primeiramente à Deus por ter me guiado e sustentado até aqui, agradeço aos meus pais Jurandir e Arlete e toda minha família que tanto se preocuparam comigo nesse tempo longe de casa, aos meus amigos, antigos e novos, de longe e de perto que estiveram comigo por toda essa caminhada. Agradeço também a Igreja Presbiteriana de Vila Celeste, sem ela não teria trilhado o caminho que trilhei. Agradeço a Universidade Federal de Ouro Preto por me proporcionar ensino de qualidade, à professora Gilda Assis, que tanto me apoiou, mesmo quando eu pensei que não ia conseguir. E a todos que pouco ou muito estiveram comigo, mesmo aqueles que me proporcionaram dificuldades, pois estas dificuldades me moldaram a ser quem eu sou hoje.

"If we find ourselves with a desire that nothing in this world can satisfy, the most probable explanation is that we were made for another world."

— C.S Lewis (1898 – 1963),
in: Mere Christianity

Resumo

Sistemas legados e sistemas desenvolvidos para fins de pesquisa muitas vezes necessitam de modificações, seja conter código que pode ser melhorado para fins de facilitação da compreensão, código duplicado ou por apresentar problemas de desempenho e de incompatibilidade com hardware e sistemas operacionais mais recentes. Por meio da utilização da refatoração, torna-se possível a modificação de um ou vários componentes do sistema a fim de se obter melhorias, sem alterar a funcionalidade percebida do sistema. O sistema de realidade aumentada NeuroR é um sistema de apoio para reabilitação de ombro após o AVC(Acidente Vascular Cerebral), que utiliza estímulos visuais, produzidos por um braço virtual que substitui o braço real com sequelas motoras na imagem adquirida em tempo real. Foi realizada uma análise estática e dinâmica no código do sistema NeuroR que serviu como guia para o processo de refatoração. Como resultado, o código fonte do NeuroR foi refatorado, se tornando mais escalável, conciso e facilitando sua manutenção. Também foram levantadas e comparadas bibliotecas de realidade aumentada que podem ser utilizados na refatorações futuras do sistema NeuroR para sua modernização, tornando-o mais robusto e escalável.

Palavras-chave: realidade aumentada, reabilitação, refatoração.

Abstract

Legacy systems and systems developed for research often require modification, either because they contain code that can be improved for ease of understanding, duplicate code, or because of performance issues and incompatibility with newer hardware and operating systems. By using refactoring, it becomes possible to modify one or more system components to achieve improvements without changing the perceived functionality of the system. The NeuroR augmented reality system is a support system for shoulder rehabilitation after stroke that utilizes visual stimuli produced by a virtual arm that replaces the real arm with motor sequelae in the acquired real-time image. A static and dynamic analysis was performed on the NeuroR system code that served as a guide for the refactoring process. As a result, NeuroR's source code has been refactored, making it more scalable, concise, and easier to maintain. We have also raised and compared augmented reality libraries that can be used in future refactorings of the NeuroR system for its modernization, making it more robust and scalable.

Key-words: augmented reality, rehab, refactoring.

Lista de ilustrações

Figura 1 – Exemplo de sobreposição de imagem real com uso de marcador (ARTOOLKIT, 2007) 15
Figura 2 – Braço virtual do NeuroR (ASSIS, 2010) 16
Figura 3 – Marcador Hiro, da ARToolKit 17
Figura 4 – Exemplo de remoção de objeto de interesse (TONIETTO; WALTER, 2000) 18
Figura 5 – Remoção do braço com o uso de chroma-key (ASSIS, 2010) 18
Figura 6 – Arquitetura do Sistema NeuroR (ASSIS, 2010) 21
Figura 7 – Exercício de abdução 22
Figura 8 – Abdução do ombro 23
Figura 9 – Adução do ombro 23
Figura 10 – Flexão do ombro 23
Figura 11 – Eletrodo no músculo deltóide (ASSIS, 2010) 24
Figura 12 – Funcionamento do ARToolKit (ASSIS, 2010) 25
Figura 13 – Estrutura de arquivos antes da refatoração 32
Figura 14 – Estrutura de arquivos depois da refatoração 33
Figura 15 – Fluxo de chamadas de funções do NeuroR 42

Lista de tabelas

Tabela 1 – Comparativo quanto as licenças 29
Tabela 2 – Comparativo quanto as plataformas suportadas 30
Tabela 3 – Comparativo quanto ao suporte de geração de marcadores 30
Tabela 4 – Antes e depois dos arquivos de cabeçalho 36
Tabela 5 – Antes e depois dos arquivos .cpp 36
Tabela 6 – Números de funções presentes antes e depois da refatoração 37

Lista de abreviaturas e siglas

API	*Application Programming Interface*
AVC	Acidente vascular cerebral
EMG	Sinal elétrico miográfico
RA	Realidade aumentada
RGB	*Red Green Blue*
RV	Realidade virtual

Sumário

1 INTRODUÇÃO 12
1.1 Identificação do problema 12
1.1.1 Objetivos propostos 13

2 REVISÃO BIBLIOGRÁFICA 14
2.1 Realidade Aumentada 14
2.1.1 Tipos de RA 14
2.1.2 Tipos de rastreamento 15
2.2 Marcadores 17
2.3 Chroma-Key 17
2.4 Refatoração 18

3 MATERIAIS E MÉTODOS 20
3.1 Sistema NeuroR 20
3.1.1 Arquitetura do NeuroR 21
3.1.2 Execução do NeuroR 24
3.2 ARToolkit 24
3.3 Métodos 25
3.3.1 Entendimento do Código 25
3.3.2 Inicio da refatoração 27

4 RESULTADOS 28
4.1 Biblioteca de realidade aumentada 28
4.1.1 Vuforia Augmented Reality 28
4.1.2 OpenCV 28
4.1.3 ARToolkit 29
4.2 Comparação Entre as Bibliotecas 29
4.3 Análise dos Arquivos do NeuroR e Mudança no Sistema de Diretórios 30
4.4 Problema Com a Ordem das Teclas 32
4.5 Remoção dos Warnings 34
4.6 Análise Estática e Dinâmica das invocações de funções no NeuroR e remoção de Código Morto 35
4.7 Renomeação das variáveis 37
4.8 Identificação de Arquivos e Funções Críticas 37
4.9 Aplicação da Extração de Método 37
4.10 Testes 41

4.11 Fluxo de execução 41
4.12 Compilação e Montagem 43
4.12.1 Criação do Makefile e Facilidade Para o Usuário 43
4.13 Recursos de Ajuda 44

5 CONCLUSÃO 45

REFERÊNCIAS 46

1 Introdução

Pacientes que sofreram um acidente vascular cerebral (AVC) muitas vezes perdem a mobilidade de um dos lados do corpo, devido a uma lesão cerebral no hemisfério contralateral. Estes pacientes devem ser submetidos a tratamentos de reabilitação (ASSIS, 2010).

O sistema de realidade aumentada NeuroR foi proposto como um sistema de apoio para a reabilitação de ombro após o AVC. Ele fornece um estímulo visual do movimento de um braço virtual, o qual substitui o braço real, correspondente ao lado paralisado, numa imagem (ASSIS, 2010).

O sistema NeuroR começou ser desenvolvido em 2009 para fins de pesquisa e não se teve uma preocupação com o design interno, reuso, otimização e documentação na implementação.

Para que o sistema NeuroR possa ser adotado com suporte na reabilitação de membro superior pelos profissionais, é necessário aplicar um processo de refatoração do código para torná-lo compatível com o hardware e software atuais, melhorar o seu desempenho, manutenibilidade e usabilidade.

Refatoração em código significa tornar o código mais compreensível e bem estruturado alternando o seu design interno de modo a facilitar o desenvolvimento em equipe e a manutenção (FOWLER, 2009).

1.1 Identificação do problema

O sistema NeuroR foi construído como um protótipo de uma pesquisa de doutorado e reflete o resultado dos vários experimentos que estavam sendo desenvolvidos de forma não sistêmica em um curto espaço de tempo. Foram implementadas diversas rotinas no NeuroR que não são mais utilizadas na versão atual.

A pesquisa envolvendo o sistema NeuroR (ASSIS, 2010) (ASSIS, 2012) demonstrou bons resultados clínicos e de conectividade cerebral, tais resultados nortearam a decisão de tornar-lo mais que uma objeto de pesquisa acadêmica, podendo ser um produto de *software* que pode auxiliar na recuperação de pacientes que tiveram sequelas na movimentação do ombro depois de um acidente vascular cerebral(AVC).

Dessa forma este trabalho visa refatorar o software NeuroR, a fim de torna-lo mais moderno, escalável, sempre visando a facilidade de uso para o usuário final(pacientes e profissionais da saúde) bem como a manutenibilidade do sistema.

1.1.1 Objetivos propostos

O objetivo principal deste trabalho consiste em através de estudos e análises do projeto NeuroR, refatorar o código, melhora-lo, torna-lo escalável e facilitar sua manutenção.

Os objetivos específicos são:

- Efetuar um levantamento de ferramentas que possam auxiliar nesses processos de *refactoring*.
- Estudar as ferramentas atuais de visão computacional e realidade aumentada.
- Efetuar uma análise estática e dinâmica profunda do código.

Para atingir esses objetivos, torna-se necessário entender melhor sobre a tecnologia de realidade aumentada, compreender e explorar o código fonte do NeuroR e refatora-lo.

2 Revisão bibliográfica

2.1 Realidade Aumentada

Realidade aumentada(AR do inglês, *augmented reality* ou RA em português) pode ser definida como sobreposição de imagens virtuais no mundo real, onde os usuários podem interagir com os artefatos virtuais (KATO; BILLINGHURST, 1999). Diferentemente da realidade virtual (RV), que transporta o usuário para um ambiente virtual, a RA mantém o usuário no seu ambiente físico e transporta elementos do ambiente virtual para o espaço do usuário, permitindo a interação com o mundo real e virtual, de maneira natural e sem necessidade de adaptações.

O sistema de realidade aumentada deve permitir a interação entre os objetos virtuais e reais, ter uma calibração precisa e registro adequado dos objetos virtuais no mundo real. Registro é o alinhamento dos objetos virtuais em posição e orientação no mundo real. A realidade aumentada mantém o usuário no seu ambiente físico e transporta os objetos sintéticos para o mundo real (TORI; KIRNER; SISCOUTTO, 2006). Um sistema completo de RA é constituído por uma ou mais câmeras, algum software para construção de objetos virtuais, sistema gráfico, dispositivo de interação e dispositivo de visualização, que pode ser monitor ou tela de LCD ou plasma, HMD (*head mounted display*) ou alguma superfície de projeção espacial. O funcionamento de um sistema de RA compreende:

- aquisição de imagens da cena real,
- criação de imagens virtuais,
- rastreamento para posicionamento e orientação espacial dos objetos virtuais,
- sobreposição/composição dos objetos virtuais no cenário real e
- interação em tempo real.

A figura 3 mostra exemplo de realidade aumentada, onde os objetos geométricos cone e esfera foram sobrepostos à imagem do mundo real.

2.1.1 Tipos de RA

A RA pode ser classificada em RA direta e RA indireta, dependendo da forma que o usuário observa o mundo misturado. Quando o usuário vê o mundo misturado apontando os olhos diretamente para as posições reais, como na RA espacial, a RA é de visão direta ou imersiva. Quando o usuário vê o mundo misturado em algum dispositivo, como uma

tela de exibição ou projeção, não alinhado com as posições reais, a RA é de visão indireta ou não imersiva (KATO; BILLINGHURST, 1999).

Quanto aos dispositivos de visualização utilizados, os tipos de sistema RA possíveis são (MILGRAM et al., 1995):

- visualização direta óptica,
- visualização direta baseada em vídeo,
- visualização indireta baseada em monitor,
- visualização indireta baseada em projetor,
- visualização direta baseada em projetor (RA espacial).

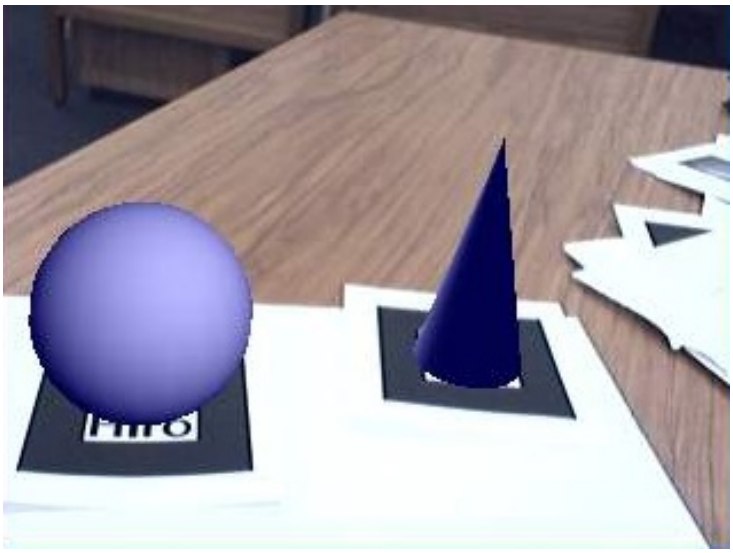

Figura 1 – Exemplo de sobreposição de imagem real com uso de marcador (ARTOOLKIT, 2007)

O NeuroR faz uso de realidade aumentada de visão indireta baseada em monitor quando sobrepõe a imagem de um braço virtual em cima do braço real do paciente. Como mostrado na Figura 2.

2.1.2 Tipos de rastreamento

Uma RA também pode ser classificada pelo seu tipo de rastreio, chamado de *tracking* (AMIN; GOVILKAR, 2015). As principais são:

- Rastreamento baseado em marcador:

  O tipo mais comum de rastreamento, efetua o rastreio através de reconhecimento de um padrão pré-conhecido pelo programa. Usando um marcador pode-se relacionar

Figura 2 – Braço virtual do NeuroR (ASSIS, 2010)

tanto os pontos no espaço quanto a distancia e angulo. Marcadores costumam ser figuras simples, com imagens geométricos em preto e branco. Falaremos especificamente de marcadores ainda.

- Rastreamento hibrido:

  Esse tipo de rastreamento geralmente combina duas ou mais fontes de dados, como GPS, bússola, acelerômetro para calcular a posição e orientação reais (BILLINGHURST, 2013). O sistema de posicionamento global permite identificar a localização atual do dispositivo, com essas informações podemos encontrar objetos em nossa área a ser aumentada. Com o uso da bússola, podemos dizer a direção do dispositivo apontando e verifique se esse caminho possui algum objeto a ser aumentado. O acelerômetro é usado para calcular a orientação do dispositivo usando a gravitação para sua vantagem.

- Rastreamento baseado em modelo:

  O rastreamento baseado em modelo, utiliza conhecimento prévio de formas objetos 3D no ambiente juntamente com sua aparência. Detecção por modelo funciona usando detecção de bordas dos objetos. Em alguns casos, o modelo é fornecido para rastrear semelhança em relação ao seu objeto no ambiente, por exemplo: rastrear um carro em movimento na rua, mas essa abordagem geralmente requer muito mais poder de processamento. (BILLINGHURST, 2013).

2.2 Marcadores

Em realidade aumentada é comum o uso de marcadores para facilitar as interações entre virtual e real. Marcadores funcionam para marcar pontos, usualmente tem padrões e cores para que seja fácil reconhece-los e são colocados no mundo real nas posições onde os objetos virtuais devem ser exibidos. Existem vários exemplos de marcadores (Hiro-AR Tool Kit, ARuco, QR Codes).

O marcador para o reconhecimento do ombro lesado no NeuroR é um dos marcadores disponibilizados com a biblioteca ARToolkit(Figura 3). O sistema NeuroR não conta com a funcionalidade de cadastrar outras imagens como marcador. Também é possível desenvolver aplicações de RA sem o uso de marcadores(*markerless*), que são mais complexas pois não tem o padrão para reconhecimento de forma prévia e tem que em tempo real encontrar os padrões(faces, mãos e etc), utilizando técnicas de aprendizado de máquina e processamento de imagens.

Figura 3 – Marcador Hiro, da ARToolKit

2.3 Chroma-Key

A técnica consiste em substituir os *pixels* de uma determinada cor por outro objeto no espaço de visualização da câmera, ou seja é uma forma de remoção de objetos de interesse. Segundo (SEMENTILLE et al., 2007) técnica de *chroma-key* se baseia na identificação de partes da imagem onde se encontra uma determinada cor, denominada "cor chave".

A Figura 4 mostra a utilização do Chroma-Key para remoção do fundo da imagem e em seguida sua substituição por outra imagem.

O paciente veste uma luva de cor predominante, no caso amarelo(escolhida como cor chave), o sistema NeuroR se orienta através do reconhecimento da cor e efetua a sobre-

Figura 4 – Exemplo de remoção de objeto de interesse (TONIETTO; WALTER, 2000)

posição do membro afetado. Para esse reconhecimento acontecer, o sistema RGB(*red, green, blue*), que possui as cores primárias vermelho, verde e azul(SÜSSTRUNK; BUCKLEY; SWEN, 1999) é usado.

A Figura 5 mostra a luva de cor predominante amarela sendo removida na tela com o chroma-key.O objeto removido (luva amarela) é substituído pela imagem de fundo, capturada previamente.

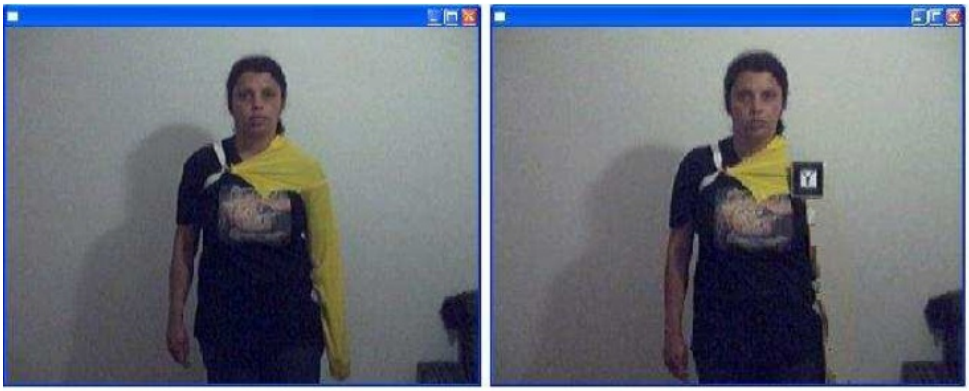

Figura 5 – Remoção do braço com o uso de chroma-key (ASSIS, 2010)

2.4 Refatoração

Segundo (FOWLER, 2009), refatoração é uma alteração feita na estrutura interna do software para torná-lo mais fácil de entender e mais barato para modificar sem alterar seu comportamento observável. Segundo (SOMMERVILLE, 2007), a reengenharia de software ou engenharia reversa se ocupa de reimplementar sistemas legados, para que sua manutenção seja mais fácil.

Ainda segundo (FOWLER, 2009) refatoração se baseia nos princípios básicos de modificar o programa em passos pequenos e que o bom código deve ser compreendido também por humanos e não somente por computadores. A refatoração é uma ferramenta que ajuda a manter os códigos mais limpos e seguros. A reengenharia pode envolver redocumentar, organizar e reestruturar o sistema, traduzir o sistema para uma linguagem

de programação mais moderna e modificar e atualizar a estrutura e os valores dos dados do sistema. A funcionalidade do software não é modificada e, normalmente, a arquitetura do sistema também permanece a mesma. O sistema NeuroR é um bom candidato ao processo de refatoração de software pois a sua documentação existente não reflete o código atual, resultante de diversas mudanças ao longo do tempo. O processo de refatoração e reengenharia do sistema NeuroR também tornará possível incorporar, de forma mais simples, novas funcionalidades para o suporte à reabilitação motora pós AVC.

O uso dessa técnica prolonga a vida de um software facilitando a manutenção e adequação as novas demandas de tecnologia, por exemplo: surgimento de novas bibliotecas.

No processo de refatoração, pode-se encontrar novas necessidades que quando o programa foi escrito não existiam, ou mudanças arquiteturais ou de ambiente de programação, como atualização do sistema operacional ou mesmo mudanças no ambiente de desenvolvimento que podem fazer com o que o código fonte deixe de funcionar necessitando assim que o processo de refatoração também inclua levantamento de novos requisitos que o tornem funcional novamente ou mesmo que seja para tornar o código mais portável ou robusto.

Para (FOWLER, 2009), algumas técnicas são primordiais para que uma refatoração ocorra, dentre elas citamos: Extrair Método, Mover Método, Mover Atributo e Renomear Método. Abaixo, uma breve descrição de cada uma delas.

Na técnica de extrair método, (FOWLER, 2009) também diz que sua refatoração ocorre quando sua principal motivação for eliminar métodos grandes que realizam duas ou mais tarefas, possuindo também muitos comentários, um método muito grande pode muitas vezes gerar um ou mais métodos novos.

A técnica de refatoração “mover método” se dá também de forma simples. Ao possuir um método que será utilizado muitas vezes por outra classe, é passível de copiar o método usado pela classe “a” e colar para a classe “b” (FOWLER, 2009).

Muito parecida com a técnica anterior, também pode ser aplicada a técnica “mover atributo”. Segundo (FOWLER, 2009) fala que pode-se aplicá-la quando uma classe “b” utiliza muito um atributo que foi definido em uma classe “a”, assim, deve-se apenas copiar o atributo da classe definida para a classe que o utiliza muitas vezes.,

A refatoração do NeuroR surge como importante, pois sendo um sistema de baixo custo, e alto potencial de impacto social, devemos melhora-lo para que ele possa estar pronto para mais melhorias e proporcionar um melhor entendimento do código para futuros programadores.

3 Materiais e Métodos

Para o desenvolvimento deste trabalho foram utilizados o código fonte do NeuroR escrito na linguagem C++, a biblioteca ARtoolKit presente no sistema, API OpenGL e o compilador GCC versão 5.1.0.

Todo o processo foi feito utilizando o sistema operacional Windows 10.

3.1 Sistema NeuroR

Acidentes vasculares cerebrais têm causado a incapacidade física em adultos e até mesmo em crianças, em todo o mundo. O Acidente Vascular Cerebral (AVC) é a terceira causa de morte em todo mundo, atrás apenas das doenças do coração e câncer. As deficiências motoras provocadas por essa patologia se caracterizam por paralisia ou fraqueza no lado do corpo oposto ao da lesão, gerando déficits no movimento do membro superior e na marcha. Segundo dados do Internet Stroke Center, AVCs causam cerca de 140 mil mortes por ano nos EUA (STROKECENTER.ORG,). A reabilitação desempenha um papel importante no sentido de corrigir ou diminuir as deficiências motoras e aumentar a independência funcional dos pacientes que sofreram AVC.

Nesse contexto surgiu a pesquisa de doutorado onde foi proposto e implementado o NeuroR, um sistema de realidade aumentada para investigar se a percepção visual da movimentação de um braço virtual tridimensional, sobreposto à imagem do paciente e conectado ao ombro substituindo o braço real lesado, pode afetar positivamente a reabilitação motora deste membro comprometido. (ASSIS, 2010), Gilda Assis, autora da tese, demonstrou que o sistema poderia ser útil para a reabilitação motora de ombro de pacientes pós-AVC, os resultados foram satisfatórios (ASSIS, 2010) e (ASSIS, 2012) e concluíram que existe aplicabilidade para o problema proposto.

Há diversas técnicas para estimular a neuroplasticidade após uma lesão encefálica, as quais constituem as terapias de reabilitação e que podem promover a reabilitação física, cognitiva e psico-social do indivíduo (NUDO; PLAUTZ; FROST, 2001)

As terapias de reabilitação física são técnicas que têm como objetivo corrigir ou diminuir as deficiências perceptuais e motoras do paciente. Várias terapias de reabilitação física têm sido utilizadas no tratamento de pacientes que sofreram um acidente vascular cerebral (AVC), como prática física, terapia ocupacional, terapia robótica, hidroterapia, musicoterapia, estimulação elétrica funcional e prática mental.

A prática mental é uma terapia de reabilitação que visa estimular a imagem motora do paciente. A imagem motora pode ser definida como um estado dinâmico no qual as

representações de uma determinada ação motora são ensaiadas na memória de trabalho sem que haja necessariamente qualquer saída motora (DECETY, 1996).

No tratamento de pacientes no estágio crônico, ou seja, que sofreram um AVC há mais de um ano, os resultados de reabilitação motora de redução no tônus muscular, recuperação da capacidade de fracionar os movimentos e uso do membro afetado nas atividades do dia-a-dia, são considerados difíceis de obter. Por esse motivo, muitos pacientes recebem alta do tratamento com as terapias de reabilitação convencionais quando completam um ano pós-AVC e também muitas empresas de planos de saúde cessam o financiamento do tratamento após o período de um ano (ASSIS, 2010).

Diante desse cenário, o sistema NeuroR foi proposto como um sistema de baixo custo que pode ser utilizado como recurso suplementar de reabilitação motora pelos indivíduos no próprio domicílio ou em ambientes de reabilitação convencionais, como clínicas e hospitais.

3.1.1 Arquitetura do NeuroR

A arquitetura do sistema é composta de oito módulos, como mostra a figura 6.

Como proposto por (ASSIS, 2010) o processo de utilização do NeuroR dá-se da seguinte maneira:

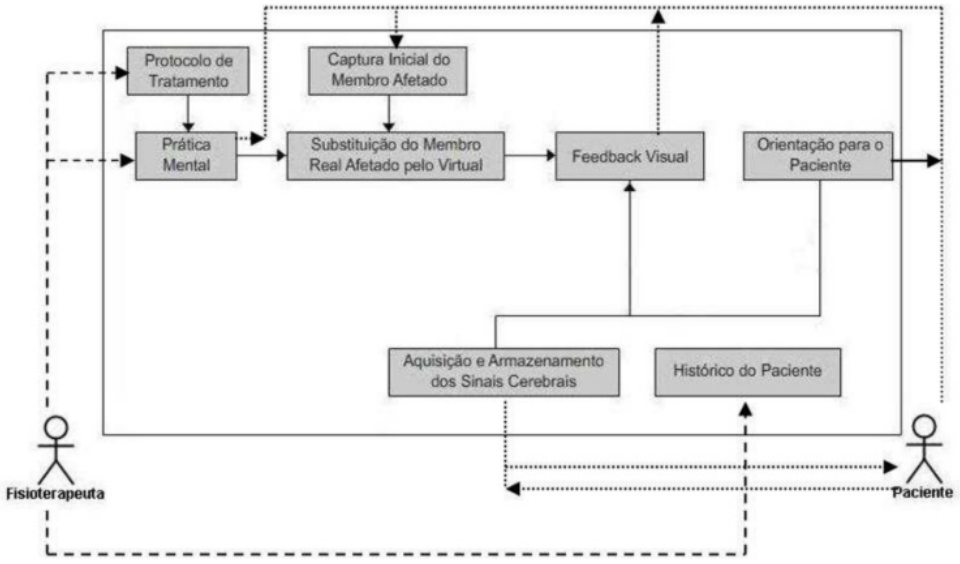

Figura 6 – Arquitetura do Sistema NeuroR (ASSIS, 2010)

Inicialmente o fisioterapeuta define a configuração do sistema, selecionando um programa de reabilitação adequado ao paciente. Durante a intervenção, o módulo de "Protocolo de Tratamento" segue o programa de reabilitação selecionando o exercício correto para a intervenção atual. O módulo de "Prática Mental" é sincronizado com o exercício selecionado do programa de reabilitação. As orientações para a prática mental

podem ser gravadas em arquivos de áudio e fornecidas pelo fisioterapeuta ao sistema. O arquivo de áudio da prática mental inicia com um relaxamento. Nos minutos finais da intervenção a prática mental reconduz o paciente ao espaço presente. As orientações para a prática mental também podem ser feitas pelo fisioterapeuta durante a intervenção, sem o uso dos arquivos de áudio. O módulo de "Aquisição e Armazenamento dos Sinais Cerebrais" está continuamente obtendo amostras de sinais EMG (eletromiografia) do músculo selecionado e armazenando-as juntamente com a informação do tempo em um arquivo padrão de dados. Ele também pode se comunicar com o módulo de *Feedback* Visual para disparar a animação do braço virtual durante a prática física. Uma câmera de vídeo captura a imagem frontal do paciente. Um marcador fiducial, impresso, é fixado no ombro do membro lesado. O módulo "Captura Inicial do Membro Afetado" utiliza um software baseado em visão computacional para o rastreamento do marcador.

O módulo de "Substituição do Membro Real Afetado pelo Virtual" remove o braço paralisado da imagem e mistura a imagem com o braço virtual. Cada quadro da câmera é processado por um computador que sobrepõe o gráfico 3D na imagem. O módulo "*Feedback* visual" mostra a imagem em RA na tela do computador. Durante a intervenção, quando a prática mental induz um exercício, o módulo "*Feedback* Visual" inicia a animação, que é disparada pelo operador do sistema, a Figura 7 mostra o exercício chamado de "abdução"sendo executado.

O módulo "Histórico do Paciente" registra as informações coletadas pelos médicos e outros profissionais de saúde que cuidaram do paciente e também o acompanhamento do bem-estar do indivíduo: assistência, fatores de risco, exercícios e perfil psicológico.

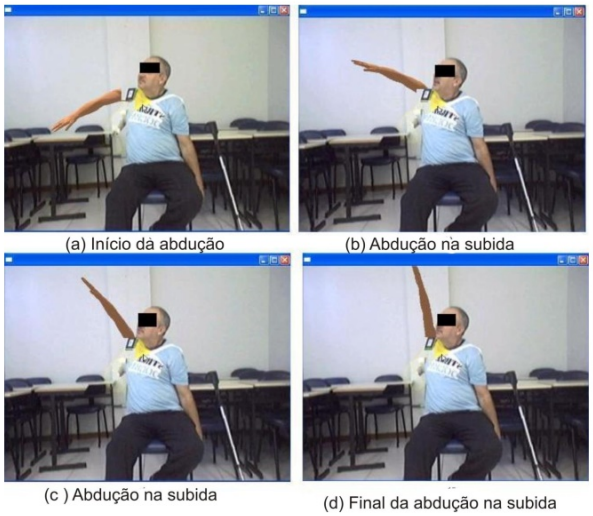

Figura 7 – Exercício de abdução

O NeuroR implementou 3 exercícios de ombro para a prática mental, os quais são: abdução do ombro, adução de ombro e flexão de ombro. As Figuras 8, 9 e 10 mostram respectivamente as primeiras partes das animações dos movimentos, a partir da posição

anatômica inicial, o resto da animação consiste na volta até a posição inicial.

Figura 8 – Abdução do ombro

Figura 9 – Adução do ombro

Figura 10 – Flexão do ombro

No módulo de prática física do NeuroR foi desenvolvido uma interface cérebro-computador para o disparo do movimento do braço virtual no módulo de Aquisição e Armazenamento dos Sinais Cerebrais.

Sinais elétricos são capturados do músculo e interpretados em tempo real através de um eletrodo posicionado no músculo deltóide médio. A Figura 11 mostra o eletrodo posicionado no músculo deltóide.

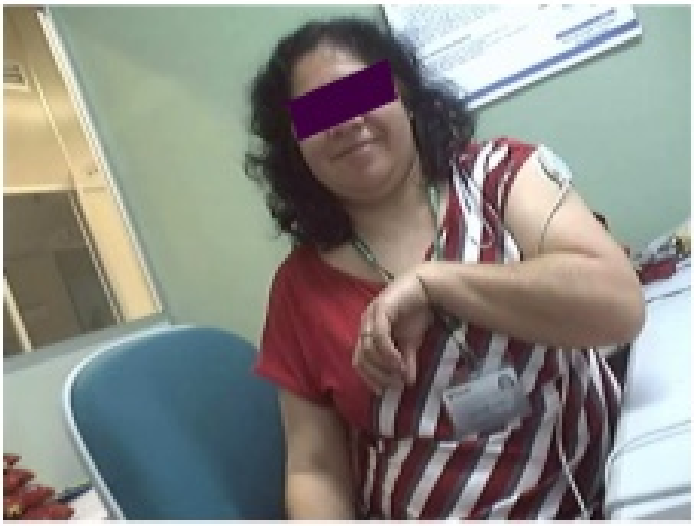

Figura 11 – Eletrodo no músculo deltóide (ASSIS, 2010)

3.1.2 Execução do NeuroR

Foram utilizados para o desenvolvimento e funcionamento do NeuroR os seguintes recursos de *hardware* e *software*:

- uma *webcam* ou câmera integrada a tela de *notebook*;
- luva amarela sem dedos;
- marcador fiducial para fixação no ombro com sequelas;
- aparelho para aquisição de EMG e eletrodos, Miotec em 2009 e depois em 2016 utilizei o MyoArmBand;
- biblioteca para acesso ao aparelho de EMG;

3.2 ARToolkit

ARtoolKit foi originalmente desenvolvida em 1999 por Hirozaku Kato no Nara Institute of Technology (KATO; BILLINGHURST, 1999) e foi lançado pela Universidade de Washington HIT Lab.

Em 2001 a versão 1.0 foi lançada, juntamente com uma versão *open-source*. A ARtoolKit foi umas das primeiras bibliotecas de realidade aumentada a ter suporte à dispositivos móveis, ainda em 2005, ele já era no sistema operacional Symbian, da Nokia, um dos primeiros projetos de *Smartphone* como conhecemos hoje.

O ARToolKit utiliza técnicas de visão computacional para calcular a posição e orientação de marcadores, identificados no cenário real, a partir das sequências de

vídeo capturados por uma *webcam*, possibilitando a sobreposição de objetos virtuais nos marcadores. A Figura 12 mostra o ciclo básico da execução da biblioteca ARToolKit. Inicialmente a imagem do mundo real é capturada por um dispositivo de entrada de vídeo para dar início à identificação dos marcadores. A imagem real capturada é transformada em imagem binária. Esta imagem é analisada em busca de regiões quadradas. Ao encontrar uma região quadrada, a ferramenta calcula a posição e orientação da câmera em relação a esta região buscando identificar figuras específicas, denominadas marcadores. Os marcadores são símbolos distintos e previamente cadastrados através de um treinamento da rede neural do ARToolKit para seu reconhecimento efetivo. Uma vez reconhecido o marcador, a ferramenta calcula o ponto em que o objeto virtual deve ocupar no mundo real e realiza a sobreposição das imagens retornando ao usuário a combinação visual do mundo real e do objeto virtual.

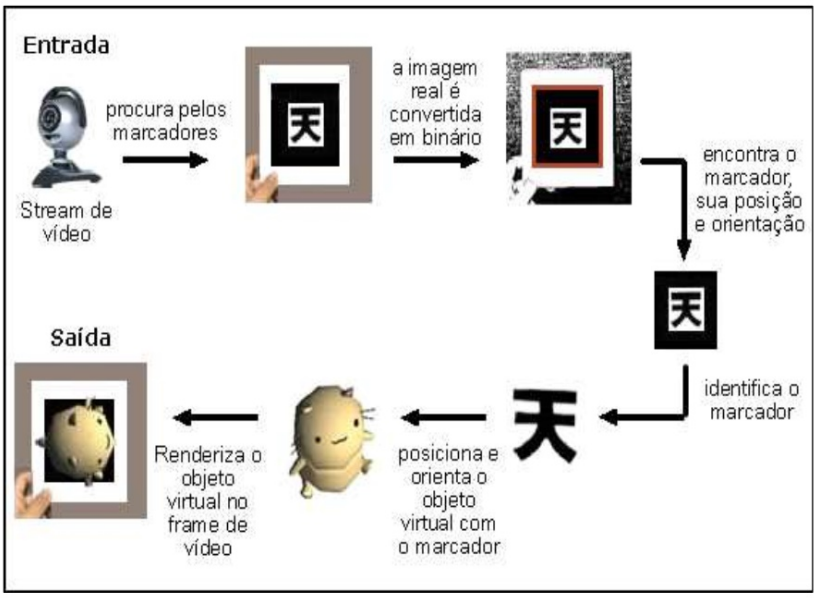

Figura 12 – Funcionamento do ARToolKit (ASSIS, 2010)

Graças ao surgimento de outras bibliotecas, seu uso foi diminuindo, hoje ela possui uma comunidade menos ativa comparada a outras bibliotecas como OpenCV e Vuforia. Ela é mantida por uma empresa chamada DAQRI que adquiriu seus direitos e deixou seu código completamente *open-source* liberando *features* que eram exclusivas da versão profissional de quando ela era mantida pela HIT lab (TECHCRUNCH,).

3.3 Métodos

3.3.1 Entendimento do Código

O NeuroR foi escrito em C++, utilizando tanto OpenGL quanto uma biblioteca de RA, a ARtoolKit como já foi dito, tais tecnologias constituem interações complexas.

A biblioteca utilizada para o processamento gráfico no NeuroR é a OpenGL que é uma API para processamento gráfico largamente usada, multi-linguagem, multiplataforma. Essa biblioteca fornece uma abstração de qualidade no que tange a processamento gráfico. Ela representa uma série de funções que podem ser chamadas pelo programa cliente, realizando as comunicações de baixo nível de manipulação de vetores 2d e 3d.

Códigos que utilizam video, costumam trabalhar como se fossem máquinas de estados e em laços constantes, isso se dá por causa da natureza de um video que é uma sucessão de quadros(*frames*) na tela.

Basicamente o NeuroR pode ser divido em três etapas que são:

- Inicialização, que inclui a declaração de variáveis e chamadas de bibliotecas
- Checagem de estados, que inclui saber em que estado o programa se encontra(se o fundo já foi capturado, se o paciente já foi capturado, se um marcador foi encontrado), nessa etapa em *loop* o programa também fica esperando por *inputs* do usuário, que são as teclas de comando.
- Execução de exercícios, se tudo está pronto para ser usado, o programa passa então para as execuções de processamento gráfico como: remoção do braço real com a luva, desenho do braço 3D e execução dos movimentos.

o sistema NeuroR usa de assincronismo(*threads*) para capturar a entrada do usuário, ou seja as teclas de comando, pois em paralelo ao laço principal do código existe um evento que fica 'escutando' a entrada de teclas. Seis teclas principais são usadas no funcionamento do NeuroR, são elas e suas funções de modo simplificado:

- **G**, para capturar a imagem sem o paciente, o fundo será usado na remoção do braço com a luva.
- **C**, captura a imagem com o paciente.
- **I**, inicia o processo de procura do marcador, neste ponto o *software* está pronto para realizar o desenho do braço virtual.
- **A**, chama o exercício 'adução'.
- **B**, chama o exercício 'abdução'.
- **F**, chama o exercício 'flexão'.

O NeuroR é um código programado de modo procedural, e não orientado a objetos.

3.3.2 Inicio da refatoração

O processo de refatoração do NeuroR iniciou com o levantamento de bibliotecas usadas no desenvolvimento do sistema, a análise das bibliotecas invocadas e quais eram efetivamente usadas. Foram coletadas informações sobre ambientes de desenvolvimento, linguagem e bibliotecas utilizadas no desenvolvimento da primeira versão do sistema NeuroR. Percebeu-se que muitos desses recursos estavam defasados, sem manutenção e em desuso.

Originalmente o sistema NeuroR foi desenvolvido numa versão antiga do DEV C++. Foi considerado migrar o projeto para uma IDE(*integrated development environment*), mais especificamente o *Microsoft Visual Studio*, uma vez que as ferramentas de refatoração para C++ pesquisadas no trabalho são integradas em IDEs. Foi realizada uma migração parcial, porém o processo foi descartado por entendermos que para permitir maior escalabilidade e não depender de um programa especifico, ter menos dependência da máquina e de programas seria adequado compilar, link-editar e executar o sistema independente de IDE. Dessa forma, apenas para fim de edição do fonte foram usados diversos programas como *Code Blocks* e Atom(editor de textos, focado em desenvolvimento de *software*). A compilação foi realizada via terminal no sistema operacional *Windows*, sem o suporte de uma IDE.

Após a análise das bibliotecas, foi realizada uma análise estática no código-fonte do NeuroR, a partir da função main(), para identificar classes e funções que não foram utilizadas. Depois mais uma análise, dessa vez mais profunda, das chamadas do NeuroR, tanto de maneira estática quanto dinâmica, com essa análise podemos perceber que havia trechos de código morto (DEBRAY et al., 2000) no sistema.

4 Resultados

4.1 Biblioteca de realidade aumentada

Bibliotecas de realidade aumentada facilitam muitas interações dentro de aplicações que usam RA, como reconhecimento de padrões, renderização das imagens sobrepostas e rastreamento(*tracking*).

Existem hoje muitas bibliotecas que auxiliam na utilização de Realidade Aumentada, o que vem de encontro com a crescente popularização de sistemas que utilizam esse tipo de tecnologia, por exemplo os filtros de câmera do Instagram, Facebook, SnapChat e etc.

Tais bibliotecas fornecem abstrações, para manipulação de dados gráficos, reconhecimento facial, de superfícies, e sobreposição de imagens.

Durante os estudos, algumas foram levantadas, instaladas e testadas, pois a hipótese com o intuito de substituir a biblioteca ARToolKit em refatorações futuras, são elas:

4.1.1 Vuforia Augmented Reality

Vuforia é uma biblioteca de realidade aumentada moderna que executa em dispositivos móveis e computadores. É um kit de desenvolvimento privado, não possui código aberto mas possui versões para uso acadêmico, por estudantes. Ela suporta diferentes tipos de alvos(marcadores, formas), 2D e 3D, também suporta reconhecimento de texto. Ela ainda disponibiliza uma ferramenta *online* para criação de marcadores.

Um aplicativo AR feito utilizando Vuforia consiste em uma câmera que captura o quadro e passa o conteúdo para o rastreador, o conversor de imagens simplesmente converte a imagem tirada pela câmera em um formato adequado para renderização e para rastreamento interno, o *tracker* que pode carregar e ativar vários conjunto de dados ao mesmo tempo, que basicamente contém os algoritmos de visão computacional que detectam e rastreiam os objetos em tempo real.(AMIN; GOVILKAR, 2015)

Ela permite apenas exportar para o sistemas Windows 10 quando se trata de computadores pessoais, e mantém uma marca d'água em todos os *frames* do vídeo gerado, ela é usada principalmente através da *engine* de jogos Unity 3D.

4.1.2 OpenCV

OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de código aberto de visão computacional e realidade aumentada, criada em 1999 pela Intel, atualmente

mantida pela Itseez (MOURA; SILVA, 2017).

OpenCV foi projetada para produzir uma infraestrutura para rápido uso de ferramentas de visão computacional (OPENCV.ORG,).

Ela foi escrita em C, e possui versões para uso em C++, Java, Python e MATLAB.

Os testes realizados com essa biblioteca foram satisfatórios, ela disponibiliza em seu *download* de arquivos muitos *templates* para diversas aplicações, como reconhecimento facial(expressões como sorriso ou tristeza por exemplo) e módulos de RA. Outra vantagem encontrada foi a disponibilidade de fóruns *online* com uma comunidade *open-source* ativa.

4.1.3 ARToolkit

ARToolKit é uma biblioteca de código aberto (ARTOOLKIT, 2007).

O ARToolKit suporta múltiplas plataformas e para a parte de renderização usa OpenGL, GLUT é usado para manipular janelas/eventos e a biblioteca de vídeo depende do *hardware*, possui uma API para cada plataforma. A API ARToolKit está disponível em C (AMIN; GOVILKAR, 2015).

A ARToolKit será mantida no projeto pois para o NeuroR ela supre suas necessidades, visto que sua remoção acarretaria num trabalho dispendioso, porém não fica descartada a sua substituição em trabalhos futuros. Entendemos que atualizar a biblioteca de realidade aumentada poderá proporcionar mais modernidade para futuras implementações.

4.2 Comparação Entre as Bibliotecas

- Baseado na Licença:

A tabela 1 demonstra as licenças: se *OpenSource*, se possui versão livre para uso e se a biblioteca dispõem de alguma versão comercial.

Biblioteca de RA		Vuforia	ArtoolKit	OpenCV
Licença	**Open Source**	Não	Sim	Sim
	Grátis	Sim	Sim	Sim
	Comercial	Sim	Sim	Não

Tabela 1 – Comparativo quanto as licenças

- Baseado na Plataforma Suportada:

Vuforia suporta Android a partir da versão 5.1, iOS a partir da versão 11, Windows somente a versão 10 (VUFORIA.COM, 2019). ARToolKit tem suporte desenvolvimento

Android, Linux, Windows e iOS. OpenCV ARToolKit tem suporte desenvolvimento Android, Linux, Windows e iOS. Abaixo a tabela 3 demonstra esses detalhes:

Biblioteca de RA		Vuforia	ArtoolKit	OpenCV
Plataforma	**iOS**	Sim	Sim	Não
	Android	Sim	Sim	Sim
	Windows	Sim	Sim	Sim

Tabela 2 – Comparativo quanto as plataformas suportadas

- Baseado na Geração de Marcadores:

A Vuforia dispõem de um sistema *online* onde qualquer imagem pode ser registrada como um marcador, o sistema faz a conversão necessária do arquivo para o modelo de reconhecimento da Vuforia. ARTToolKit também possui um sistema semelhante, contudo com mais limitações, o marcador precisa ser quadro, deve ter borda contínua. Em seu sistema de geração de marcador o usuário precisa primeiro criar e imprimir seu marcador e com o auxilio da ferramente criar seu próprio marcador que deve obedecer algumas regras(AMIN; GOVILKAR, 2015).

Biblioteca de RA	Vuforia	ArtoolKit	OpenCV
Geração de marcadores	Gerador Online	Gerador Online	Não suporta geração online
		Provê um conjunto de marcadores pré-definidos	Provê um conjunto de marcadores pré-definidos

Tabela 3 – Comparativo quanto ao suporte de geração de marcadores

4.3 Análise dos Arquivos do NeuroR e Mudança no Sistema de Diretórios

Foram realizados testes empíricos, por tentativa e erro, para identificar as bibliotecas necessárias à montagem e execução do NeuroR. A partir dos resultados experimentais, foram removidos os seguintes arquivos de cabeçalho e libs externos ao NeuroR (ARToolkit e Opengl) dos arquivos-fonte do NeuroR, pois os mesmas não estavam sendo utilizadas, ou seja: eram incluídos mas não eram usados.

- AR/armulti.h
- AR/gsubtutil.h
- AR/matrix.h
- AR/video
- GL/gl.h,
- GL/glex.h
- GL/glu.h

O arquivo de cabeçalho Time.h também foi removido, pois não estava sendo utilizado.

Foi criado um diretório somente com as bibliotecas(libs) necessárias, depois dessa análise.

A montagem e execução do sistema NeuroR continuou funcionando normalmente.

Todas as funcionalidades do NeuroR que envolvem a interação humano-computador foram testadas como teste caixa preta, e os resultados satisfatórios.

A estrutura de arquivos ficou mais clara e concisa após a criação de pasta '*headers*' para os arquivos de cabeçalho, separando-os dos arquivos de extensão .cpp, tudo isso colocado no mesmo diretório, dessa forma facilitando a organização e manutenção. Essa restruturação exigiu mudança nas inclusões de todas os arquivos .h em todos os arquivos, adicionando a nova pasta ao caminho do arquivo, sendo feito de forma manual em cada arquivo.

Os dois trechos de código, demonstra a mudança que teve de ser feita em todos os arquivos do projeto.

Antes:

```
...
#include "Simulation.h"
#include "Loader.h"
...
```

Depois:

```
...
#include "headers/Simulation.h"
#include "headers/Loader.h"
```

. . .

A Figura 13 mostra a estrutura dos arquivos do projeto antes da reestruturação proposta. A Figura 14 apresenta a estrutura após a organização em diretórios de cabeçalho, código-fonte, modelos e bibliotecas.

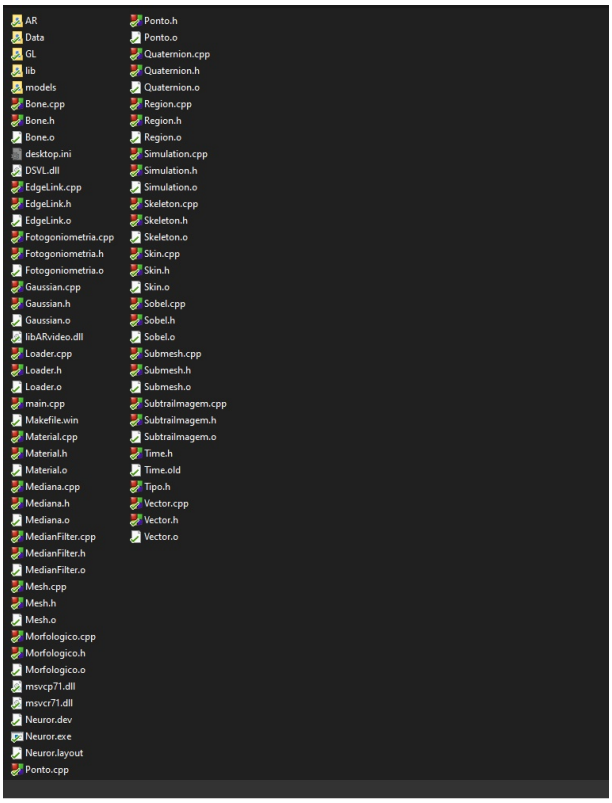

Figura 13 – Estrutura de arquivos antes da refatoração

Saímos de um ambiente de mais difícil assimilação para um ambiente mais simples.

4.4 Problema Com a Ordem das Teclas

Durante o desenvolvimento de um *software* é virtualmente impossível que o mesmo não contenha problemas, erros, os chamados *bugs*, um *bug* pode ser definido como um erro, falha no programa gerando um comportamento não desejado no funcionamento do programa.

Como explicado na seção 3.3.1, algumas das teclas que devem ser usadas no funcionamento do NeuroR são 'G','C' e 'I'. Durante o estudo e testes do sistema, foi percebido um problema com a ordem das teclas, caso a tecla 'C'(captura do fundo com o paciente) fosse apertada antes da tecla 'G'(captura do fundo sem o paciente) ou a tecla 'I' fosse apertada antes da tecla 'G' e 'C', ocorre um *crash* na tela, não permitindo que usuário efetue mais nenhuma ação e o video fica congelado ou seja a ordem 'G','C','I' precisa ser respeitada.

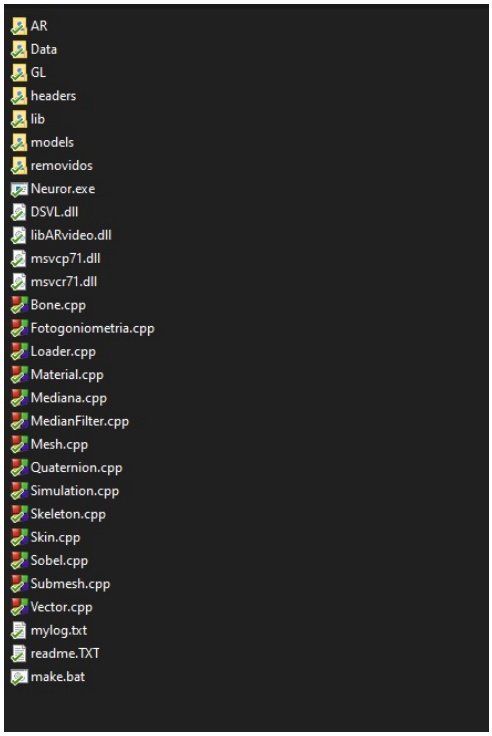

Figura 14 – Estrutura de arquivos depois da refatoração

Para eliminar este comportamento indesejado, foram feitos ajustes para fazer com que as teclas 'C' e 'I' quando apertadas fora de ordem não tenham impacto no programa.

Criamos *'flags'* para cada tecla, dessa forma foi possível conferir a legalidade da próxima tecla, assim eliminando este *bug*.

Abaixo trechos necessários para a resolução deste *bug*.

```
...
int inputG = 0;
int inputC = 0;
...
void keyEvent( unsigned char key, int x, int y){
...
case 'g':
case 'G':  //Fazer a copia da imagem de fundo
{
    inputG = 1;//flag para a ordem das teclas

    flagCapture2 = 1;
    flagVideo = 0;

    break;
}
...
case 'c':
case 'C':  //Fazer a copia da imagem do video com o paciente
{
    if(inputG == 1){ //checa se G foi apertado
        inputC = 1;

        flagCapture = 1;
```

```
            flagVideo = 0;
        }

        break;
    }
    ...

    case 'i':
    case 'I':  //Captura a imagem com o paciente e o fundo sem marcador
    {
        if(inputG==1 && inputC ==1){
            flagCapturaPacienteFundo = 1;
            flagVideo = 0;

            inputG =0;
            inputC = 0;
        }

        break;
    }
    ...

    }
```

4.5 Remoção dos Warnings

Foram realizadas alterações nos códigos-fonte para "remover"os *warnings* do processo de compilação do NeuroR. As *warnings* surgiram por causa de conversões implícitas de *string* para ponteiros de caracteres(char) que são desaconselhadas(*deprecated*) pelos compiladores novos, pois as mesmas podem resultar em resultados duvidosos. Para resolver o problema foi feito uma conversão explicita de tipos(*cast*).

Abaixo um exemplo do trecho de código que produzia um *warning*:

```
...
char *patt_name1 = "Data/patt.hiro";
...
```

E então foi mudado para:

```
...
char *patt_name1 = (char *)"Data/patt.hiro";
...
```

Onde esse tipo de coisa ocorria, foi aplicado essa mudança.

4.6 Análise Estática e Dinâmica das invocações de funções no NeuroR e remoção de Código Morto

Em ciência da computação existe o conceito de código morto(*dead code*), isto é, trechos de código inacessível, que nunca serão executados (DEBRAY et al., 2000), todavia, a existência de código morto no código fonte pode desperdiçar recursos ou ocasionar erros. Eles podem ocorrer por vários motivos, e podem ser muitas vezes condicionais a determinadas circunstancias, encontra-los muitas vezes não pode ser feito de maneira direta, e exige entendimento do problema e do fluxo de execução do programa.

Como parte da refatoração foi realizado um minucioso trabalho de entender a hierarquia de chamadas do NeuroR. Existiam diversas funções presentes no código que foram declaradas mas nunca eram chamadas. Cada função teve sua invocação acompanhada, se ela ocorria em algum ponto do código. As funções que não eram chamadas criavam seções de código morto e foram removidas, bem como suas dependências(isto é, bibliotecas que eram incluídas para o funcionamento dessas funções). Como esse processo podia ocasionar erros, sempre era verificado se a compilação continuava a ocorrer sem erros ou avisos, depois o processo era repetido. Dessa forma foram eliminadas as funções que deixaram de ser chamadas ao fim da primeira checagem. O processo foi repetido até perceber-se que o comportamento do programa permaneceu inalterado e nenhuma função presente estava sem ser invocada durante a execução.

Para gerar uma pilha de chamadas de funções foi implementado uma nova função que foi nomeada *traceToFile* que manda para um arquivo externo todas as chamadas de funções numa execução, para realizar uma análise dinâmica do código em execução.

Somando essas duas análises ao todo 9 funções foram removidas do módulo principal do código. Também foi feita uma análise dos tipos não primários usados, que exigiam uma inclusão de cabeçalhos, algumas dessas variáveis também eram declaradas mas nunca usadas, elas foram removidas bem como seus respectivos arquivos de cabeçalho.

Funções que estavam presentes no código mas nunca eram usadas:

- calculaDilatacao
- convertCal3dtoOpengl
- desenhaAlphaMap
- desenhaSobel
- drawModelSimplesBraco
- edgeLinking

- loadTexture
- pointdetection
- setModelSimplesBraco

Algumas dessas funções tiveram utilidades planejadas no projeto mas acabaram não tendo utilidade no código final do código, algumas delas não estavam terminadas.

Arquivos .cpp e .h removidos depois dessas análises foram:

- Gaussian
- Region
- EdgeLink
- SubtraiImagem
- Morfologico
- Ponto

A tabela 4 mostra a diferença entre o número de arquivos de cabeçalho existentes no código antes e depois da refatoração e sua diferença em pontos percentuais. Todos esses arquivos puderam ser removidos do código fonte sem mudar a funcionalidade do programa.

	Arquivos .h
Antes da refatoração	22
Depois da refatoração	15
Diferença	32%

Tabela 4 – Antes e depois dos arquivos de cabeçalho

A tabela 5 mostra a diferença entre o número de arquivos fonte C++ existentes no código antes e depois da refatoração e sua diferença em pontos percentuais.

	Arquivos .cpp
Antes da refatoração	21
Depois da refatoração	14
Diferença	33%

Tabela 5 – Antes e depois dos arquivos .cpp

A tabela 6 mostra a diferença entre o número de funções existentes no módulo principal antes e depois da refatoração(excluindo as funções que foram adicionadas por refatoração) e sua diferença em pontos percentuais.

	Funções
Antes da refatoração	33
Depois da refatoração	24
Diferença	27,3%

Tabela 6 – Números de funções presentes antes e depois da refatoração

As tabelas anteriores mostraram a diferença em pontos percentuais gerada pelo processo de refatoração, mantendo a mesma funcionalidade. Com menos arquivos e funções evita-se um código confuso e um ambiente de desenvolvimento de difícil assimilação, auxiliando na escalabidade e manutenção do código.

4.7 Renomeação das variáveis

Alcançar um bom nível de abstração e clareza em um código passa também pela nomeação de seus atributos, pois estas precisam possuir clareza e significado e serem explicitas em seu sentido (MARTIN, 2009).

Alguns benefícios dessa pratica consistem em:

- se numa equipe maior, a comunicação entre os membros é melhorada
- aumenta a clareza e a previsibilidade do código

As variáveis que não possuíam nomes semânticos foram renomeadas para refletir clareza ao código, dessa forma aumentando a manutenibilidade do mesmo.

4.8 Identificação de Arquivos e Funções Críticas

O sistema NeuroR possui um arquivo chamado *Simulation.cpp*, onde o a função *main()* está presente, este é muito maior que os demais, e funciona como o módulo principal do sistema, esse arquivo é um candidato a se subdividir em vários outros para trabalhos futuros.

4.9 Aplicação da Extração de Método

A função *mainLoop()* possui muitas atribuições e responsabilidades, por isso foi escolhida para ser refatorada, aplicamos a extração de método, isto é fragmenta-la em funções menores (FOWLER, 2009).

Ao todo 5 métodos foram extraídos da função *mainLoop()*, foram nomeados mediante sua utilidade, são eles: *checaVideo, checaCapturaLuva, checaCapturaFundo, checaCapturaTeclaC, capturaSemMarcador, histograma.*

Nesse processo não se faz simplesmente o encapsulamento de funções, mas também a adaptação do restante do código para funcionar adequadamente com as novas chamadas de funções, como identificação do escopo das variáveis(se interna, externa ou global).

Abaixo, o trecho de código contido na função *mainLoop()* sem refatoração, depois com refatoração.

```
ARUint8  *dataPtr;//ARUint8 = unsigned char
ARUint8  *buf;

int              marker_num;
int              i, j, k, kHiro, kKanji;
int tamanho;


/* grab a video frame */
if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL ) {
     arUtilSleep(2);
     return;
}
if( count == 0 ) arUtilTimerReset();
   count++;

/*if (count % 10 == 0)
     processouImagem = false;
*/
if (flagVideo == 1) {
        argDrawMode2D();
        argDispImage( dataPtr, 0,0 );
        arVideoCapNext();
        argSwapBuffers();
}
if (flagCaptureLuva == 1) { //teste
      double vertice[4][2];
       vertice[0][0] = 270; //x esquerdo superior
       vertice[0][1] = 190; //y esquerda superior
       vertice[1][0] = 400; //x direita superior
       vertice[1][1] = 190; //y direita superior
       vertice[2][0] = 400; //x direita inferior
       vertice[2][1] = 290; //y direita inferior
       vertice[3][0] = 270; //x esquerda inferior
       vertice[3][1] = 290; //y esquerda inferior


       glColor3f( 0.0, 1.0, 0.0 );
       argDrawSquare(vertice,0,0);
}


if (flagCapture2 == 1) { //captura imagem do fundo, sem o paciente
        //Fazer o flip horizontal na copia para obter imagem espelho
        tamanho = img1.linhas*img1.colunas;
        if (imgFundo.pixels == NULL) {
                imgFundo.pixels = (unsigned char *)malloc( sizeof(unsigned char) * (
                    tamanho*NUMEROCOMPONENTESCOR));
        }
        //Fazer o flip horizontal na copia para obter imagem espelho
```

```
            int aux = 0;
            for (int j=0; j < img1.colunas; j++) {//varredura de cima para baixo
               for (int i=0; i < img1.linhas; i++) {//varredura da esquerda para direita
                   k = (j*(img1.linhas) + i)*NUMEROCOMPONENTESCOR;
                   aux = (1 + j )*(img1.linhas)*NUMEROCOMPONENTESCOR; //ultimo pixel da
                       linha j
                   aux = aux - i*NUMEROCOMPONENTESCOR;
                   imgFundo.pixels[aux] = dataPtr[k]; //r
                   imgFundo.pixels[aux+1] = dataPtr[k+1]; //g
                   imgFundo.pixels[aux+2] = dataPtr[k+2]; //b
                   imgFundo.pixels[aux+3] = dataPtr[k+3]; //a
               }//for
            }//for
            printf("Acabei a copia da imagem - tecla g\n");
            flagCapture2 = 0;
            flagCapture = 0;
                capturouFundo = true;
            flagVideo = 1;
            arVideoCapNext();
            argSwapBuffers();
    }//if flagCature2 (Imagem de Fundo)

    if (flagHistograma == 1) {//capturar retangulo da imagem para o histograma
            imgRetangulo.linhas = 100;
            imgRetangulo.colunas = 130;
            tamanho = imgRetangulo.linhas * imgRetangulo.colunas;
            if (imgRetangulo.pixels == NULL) {
                 imgRetangulo.pixels = (unsigned char *)malloc( sizeof(unsigned char) * (
                     tamanho*NUMEROCOMPONENTESCOR));
            }
            int k = 0;
            for (int j = 190; j < 190 + imgRetangulo.colunas; j++) { //retangulo:
                (270,190) - (400, 190) - (400, 290) (270, 290)
                  for (int i=270; i< 270 + imgRetangulo.linhas; i++) {
                    imgRetangulo.pixels[k] = dataPtr[i*imgRetangulo.colunas + j]; //rgba
                    imgRetangulo.pixels[k+1] = dataPtr[i*imgRetangulo.colunas + j + 1];
                    imgRetangulo.pixels[k+2] = dataPtr[i*imgRetangulo.colunas + j + 2];
                    k+=NUMEROCOMPONENTESCOR;
                  }
            }//for
            printf("Acabei a copia da imagem dentro do retangulo - tecla h\n");
           // flagCapture = 0;
           // flagCaptureMarker = 1;
    }//if flagHistograma


    if ((flagCapture == 1) ) {//capturar imagem a cada frame apos tecla C
        if (flagDesenhaLinha == 0) {
                tamanho = img1.linhas*img1.colunas;
                if (img1.pixels == NULL) {
                       img1.pixels = (unsigned char *)malloc( sizeof(unsigned char) * (
                           tamanho*NUMEROCOMPONENTESCOR));
                }
                //printf ("Acabei de alocar espaco para img1 %d %d %x\n", img1.linhas,
                    img1.colunas,img1.pixels);
                //Fazer o flip horizontal na copia para obter imagem espelho


                int aux = 0;
                for (int j=0; j < img1.colunas; j++) {//varredura de cima para baixo
```

```
                    for (int i=0; i < img1.linhas; i++) {//varredura da esquerda para
                        direita
                k = (j*(img1.linhas) + i)*NUMEROCOMPONENTESCOR;
                aux = (1 + j )*(img1.linhas)*NUMEROCOMPONENTESCOR ; //ultimo pixel
                    da linha j
                aux = aux - i*NUMEROCOMPONENTESCOR;
                img1.pixels[aux]   = dataPtr[k];
                img1.pixels[aux+1] = dataPtr[k+1];
                img1.pixels[aux+2] = dataPtr[k+2];
                img1.pixels[aux+3] = dataPtr[k+3];
            }//for
            }//for
        // printf("Terminei a copia \n");
    }
}//if (flagCapture)

if (flagCapturaPacienteFundo == 1) {//captura imagem do paciente sem marcador
        tamanho = img1.linhas*img1.colunas;
            if (img2.pixels == NULL) {
                img2.pixels=(unsigned char *)malloc( sizeof(unsigned char)*(tamanho*
                    NUMEROCOMPONENTESCOR));
        }
        //Fazer o flip horizontal na copia para obter imagem espelho
        int aux = 0;
        for (int j=0; j < img1.colunas; j++) {//varredura de cima para baixo
           for (int i=0; i < img1.linhas; i++) {//varredura da esquerda para direita
               k = (j*(img1.linhas) + i)*NUMEROCOMPONENTESCOR;
               aux = (1 + j )*(img1.linhas)*NUMEROCOMPONENTESCOR ; //ultimo pixel da
                   linha j
               aux = aux - i*NUMEROCOMPONENTESCOR;
               img2.pixels[aux] = dataPtr[k]; //r
               img2.pixels[aux+1] = dataPtr[k+1]; //g
               img2.pixels[aux+2] = dataPtr[k+2]; //b
               img2.pixels[aux+3] = dataPtr[k+3]; //a
           }//for
        }//for
        //printf("Terminei a copia da imagem do paciente + fundo (img2) \n");
            flagCapturaPacienteFundo = 0;
}
```

Como podemos ver, esse trecho do *mainLoop* é um trecho extenso e de difícil assimilação. Após a refatoração, o mesmo trecho ficou da seguinte forma:

```
ARUint8 *dataPtr;
ARUint8 *buf;

int marker_num;
int i, j, k, kHiro, kKanji;
int tamanho;


/* grab a vide frame */
if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL )
{
    arUtilSleep(2);
    return;
}
```

```
if( count == 0 )
    arUtilTimerReset();
count++;

/*if (count % 10 == 0)
     processouImagem = false;
*/
checaVideo(dataPtr); //metodo gerado atraves de refatoracao

checaCaptureLuva(); //metodo gerado atraves de refatoracao

checaCapturaFundo(tamanho, k, dataPtr); //metodo gerado atraves de refatoracao TECLA
    G

histograma(tamanho, dataPtr); //metodo gerado atraves de refatoracao

checaCapturaTeclaC(tamanho,dataPtr,k); //metodo gerado atraves de refatoracao

capturaSemMarcardor(tamanho,k, dataPtr); //metodo gerado atraves de refatoracao
```

Ambos os códigos efetuam as mesmas tarefas, porém o código refatorado tem ganhos em legibilidade e manutenção, fica muito mais fácil localizar o trecho de código referido, o nome do método também visa auxiliar no entendimento.

4.10 Testes

Todas as etapas que envolveram mudança de local de arquivos ou mudança interna do código fonte foram testadas para garantir que não houve mudança no comportamento externo do *software*.

Os testes envolveram conferir se os 3 movimentos: adução, abdução e flexão ocorriam normalmente, juntamente com análise também do fluxo de execução que será abordado na próxima secão.

4.11 Fluxo de execução

O NeuroR possui 3 tipos principais de exercícios implementados, adução, abdução e flexão.

Foi gerado o fluxo de chamada de funções padrão do NeuroR, o *log* foi construído utilizando uma função que escreve num arquivo externo qual função foi chamada(seção 4.6), após análise do arquivo, o fluxo foi gerado.

A Figura 15 contém o fluxo de chamadas de função do NeuroR:

Os quadros em vermelho representam as funções, um quadro em vermelho dentro de um maior significa que essa função tem sua chamada dentro do fluxo de execução da

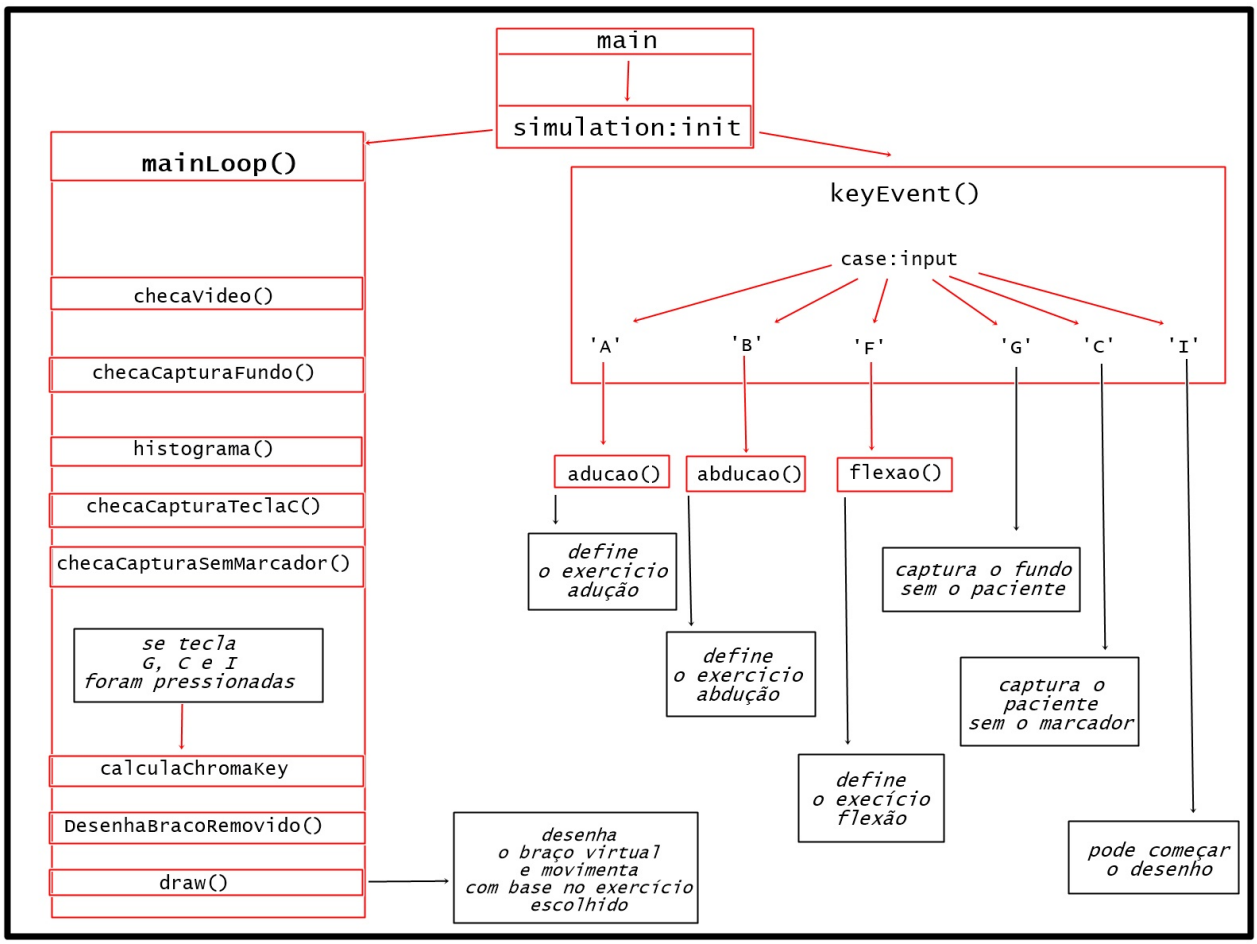

Figura 15 – Fluxo de chamadas de funções do NeuroR

de quadro maior, as funções são executadas em sequência, de cima para baixo no fluxo de leitura. Os quadros de cor preta representam comentários.

A função *main()* inicia com uma inicialização da estrutura Simulation, seu construtor e sua função seu bloco *init()*, que são semelhantes a construtores em C++.

Como podemos ver na imagem acima, a função *mainLoop()* constitui o principal fluxo de chamadas do NeuroR, juntamente com a chamada paralela da função *keyEvent()*, esta última corre de forma paralela, utilizando uma *thread* no sistema, dessa forma, o evento fica 'escutando' a entrada do usuário.

As funções geradas a partir do método de (FOWLER, 2009) 'Extrair Método' já estão incluídos no fluxo.

O objetivo do fluxograma não é dar detalhes da implementação de cada função mas dar um panorama global de como o NeuroR funciona e tem suas chamadas executadas, para inclusive auxiliar outros programadores/pesquisadores que por ventura venham a estudar/programar o NeuroR.

4.12 Compilação e Montagem

Para que a compilação do sistema computacional NeuroR se tornasse independente da IDE utilizada, foi selecionado o compilador g++ 5.1.0 sendo executado diretamente no *Command Prompt* do sistema operacional *Windows*.

Comando usado para compilar:

```
g++ -c -I. Vector.cpp Bone.cpp EdgeLink.cpp Fotogoniometria.cpp Gaussian.cpp
Material.cpp Mediana.cpp MedianFilter.cpp Mesh.cpp Morfologico.cpp Ponto.cpp
Quaternion.cpp Region.cpp Simulation.cpp Skeleton.cpp Skin.cpp Sobel.cpp
Submesh.cpp SubtraiImagem.cpp Loader.cpp
```

ou

```
g++ -c -I. *.cpp
```

O comando 'g++' invoca o compilador, esse comando pode receber argumentos, nesse caso ele recebe o argumento '-c' que o comando de compilação. O comando seguinte é o comando de inclusão, '-I.', que diz que ele deve procurar as inclusões de biblioteca no próprio diretório, em seguida o argumento '*.cpp' diz que todos os arquivos que sejam de extensão 'cpp'(da linguagem C++) devem ser incluídos na compilação. A saída gerada por esse comando são os chamados arquivos objetos, que possuem extensão '.o'.

Após a compilação, o próximo comando deve ser para efetuar a montagem e geração de um executável:

```
g++ *.o -o Neuror.exe -Llib -lARgsub -lglut32 -lglu32 -lopengl32 -lARmulti
-lAR -lgdi32 -lwinmm -lARvideo
```

Invocamos novamente o compilador com o comando 'g++' e em seguida dizemos em que arquivos ele deve operar, agora estamos interessados nos arquivos de extensão '.o' com o argumento '*.o', sem seguida dizemos o nome do arquivo de saída, que nesse caso é 'Neuror.exe'. O comando '-L' é o comando de inclusão de bibliotecas(libraries), 'lib' é o nome do diretório, que nesse caso faz parte do diretório raiz do projeto, em seguida todos os arquivos de biblioteca são listados com o argumento '-I' na frente.

4.12.1 Criação do Makefile e Facilidade Para o Usuário

Os comandos para compilação e montagem foram encapsulados num arquivo *make*, de fácil uso. O arquivo uma vez executado realiza a compilação, montagem, além de remover os arquivos objeto gerados no processo.

A saída do processo é um arquivo executável, de extensão '.exe'. A ideia foi tornar fácil a manipulação do fonte e criar uma abstração de quais bibliotecas externas usar, tornando assim a manutenção mais fácil.

O arquivo executável precisava de alguns arquivos .dll(*dinamyc link library* ou bibliotecas de ligação dinâmica) para funcionar. Como parte da pesquisa, tais arquivos foram levantados, e incluídos no pasta raiz do projeto. Esses arquivos são necessários para a comunicação do sistema operacional e o *software*, antes da refatoração tais arquivos estavam dispersos e não havia certeza de quais eram necessários.

Dessa forma, o usuário final(profissionais de saúde ou mesmo pacientes) não vão precisar de nada mais do que apenas o arquivo executável com os arquivos de .dll já levantados.

4.13 Recursos de Ajuda

Um arquivo 'readme.txt' também foi incluído, contendo os comandos necessários para a operação do software para auxiliar os usuários.

```
    Comandos
-------------------------------------------------
Selecione o braço lesionado apertando E(esquerdo) ou D(direito)
quando o programa perguntar.
Configurar vídeo para 640x480
tecla g  ----> captura imagem de fundo, com a cadeira no local
mas sem o paciente
tecla c -----> inicia a captura do paciente
tecla i  ----> captura imagem do paciente sentado na cadeira sem
a luva e sem o marcador
-------------------------------------------------
É importante que as teclas sejam apertadas nessa ordem.
Vestir a luva e com o marcador no ombro (colocar a luva primeiro
e por último o marcador mas tampá-lo com a mão até que ele esteja
bem posicionado)

-------------------------------------------------
tecla a -----> adução
tecla b -----> abdução de ombro de 0o a 180o
tecla f -----> flexão de ombro de 0o a 180o
```

Referências bibliográficas do NeuroR também foram incluídas no arquivo.

5 Conclusão

Visto a grande potencialidade social e o baixo custo que possui o sistema NeuroR na reabilitação de pacientes pós-AVC, conclui-se que é de grande valia a melhoria do *software*, adequando-o para execução e manutenção nos equipamentos, arquiteturas e sistemas operacionais disponíveis atualmente.

Foram levantadas informações sobre ambiente de desenvolvimento, linguagem e bibliotecas utilizadas no desenvolvimento da primeira versão do sistema NeuroR. Percebeu-se que muitos desses recursos estavam defasados, sem manutenção e/ou em desuso. Alguns testes para avaliar como o código fonte se comportava nos ambientes de desenvolvimento atuais foram realizados. Uma migração do NeuroR para IDEs atuais foi iniciada mas sem sucesso, ferramentas de auxilio em refatoração foram pesquisadas, porém descartadas pois as mesmas estavam integradas em IDEs, por esse motivo a refatoração foi feita de forma manual.

O sistema passou por uma refatoração de seu código, tendo ganhos principalmente em clareza e manutenção, eliminando o que não era necessário para seu funcionamento, eliminando *bugs*, melhorando a arquitetura/organização do sistema e uma refatoração interna de seu código, tendo ganhos em manutenção e clareza. Também foi gerado um arquivo *makefile*, produzido artefatos de apoio como um fluxo de execução que poderá auxiliar futuros programadores, bem como uma produção de um arquivo *readme* de instruções de uso.

Com essas mudanças esperamos que o NeuroR esteja apto para um uso em maior escala, auxiliando na recuperação de pacientes que sofreram um AVC.

Referências

AMIN, D.; GOVILKAR, S. Comparative study of augmented reality sdks. *International Journal on Computational Science & Applications*, v. 5, n. 1, p. 11–26, 2015. Citado 4 vezes nas páginas 15, 28, 29 e 30.

ARTOOLKIT. *ARToolKit*. http://www.hitl.washington.edu/artoolkit: [s.n.], 2007. Citado 3 vezes nas páginas 7, 15 e 29.

ASSIS, G. A. *NeuroR-sistema de apoio à reabilitação dos membros superiores de pacientes vítimas de acidentes vasculares encefálicos.* Tese (Doutorado) — Universidade de São Paulo, 2010. Citado 8 vezes nas páginas 7, 12, 16, 18, 20, 21, 24 e 25.

ASSIS, G. A. *Inovação Tecnológica na Saúde: edição 2012 do Prêmio Mercosul de Ciência e Tecnologia*. Tese (Doutorado) — Universidade de São Paulo, 2012. Citado 2 vezes nas páginas 12 e 20.

BILLINGHURST, M. *Introduction to Augmented Reality*. https://www.slideshare.net/marknb00/2013-426-lecture-1-introduction-to-augmented-reality: [s.n.], 2013. Citado na página 16.

DEBRAY, S. K. et al. Compiler techniques for code compaction. *ACM Transactions on Programming languages and Systems (TOPLAS)*, ACM, v. 22, n. 2, p. 378–415, 2000. Citado 2 vezes nas páginas 27 e 35.

DECETY, J. The neurophysiological basis of motor imagery. *Behavioural brain research*, Elsevier, v. 77, n. 1-2, p. 45–52, 1996. Citado na página 21.

FOWLER, M. *Refatoração: Aperfeiçoamento e Projeto*. [S.l.]: Bookman Editora, 2009. Citado 5 vezes nas páginas 12, 18, 19, 37 e 42.

KATO, H.; BILLINGHURST, M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: IEEE. *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. [S.l.], 1999. p. 85–94. Citado 3 vezes nas páginas 14, 15 e 24.

MARTIN, R. C. *Clean code: a handbook of agile software craftsmanship*. [S.l.]: Pearson Education, 2009. Citado na página 37.

MILGRAM, P. et al. Augmented reality: A class of displays on the reality-virtuality continuum. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Telemanipulator and telepresence technologies*. [S.l.], 1995. v. 2351, p. 282–292. Citado na página 15.

MOURA, G. M.; SILVA, R. L. D. S. D. Analysis and evaluation of feature detection and tracking techniques using open cv with focus on markerless augmented reality applications. *J. Mobile Multimedia*, v. 12, n. 3&4, p. 291–302, 2017. Citado na página 29.

NUDO, R. J.; PLAUTZ, E. J.; FROST, S. B. Role of adaptive plasticity in recovery of function after damage to motor cortex. *Muscle & Nerve: Official Journal of the American*

Association of Electrodiagnostic Medicine, Wiley Online Library, v. 24, n. 8, p. 1000–1019, 2001. Citado na página 20.

OPENCV.ORG. *About OpenCV*. https://opencv.org/about.html: [s.n.]. Citado na página 29.

SEMENTILLE, A. C. et al. A utilização da técnica de chromakey para composição de cenas em ambientes de realidade aumentada. In: *Proceedings-IX Symposium on Virtual and Augmented Reality-SVR*. [S.l.: s.n.], 2007. v. 1, p. 207–216. Citado na página 17.

SOMMERVILLE, I. *Engenharia de Software*. 8th. ed. [S.l.]: São Paulo: Pearson Addison-Wesley, 2007. Citado na página 18.

STROKECENTER.ORG. *Stroke Statistics*. http://www.strokecenter.org/patients/about-stroke/stroke-statistics/: [s.n.]. Acesso em: 27 nov 2019. Citado na página 20.

SÜSSTRUNK, S.; BUCKLEY, R.; SWEN, S. Standard rgb color spaces. In: SOCIETY FOR IMAGING SCIENCE AND TECHNOLOGY. *Color and Imaging Conference*. [S.l.], 1999. v. 1999, n. 1, p. 127–134. Citado na página 18.

TECHCRUNCH. *DAQRI Acquires AR Pioneer ARToolworks*. https://techcrunch.com/2015/05/13/daqri-acquires-ar-pioneer-artoolworks/: [s.n.]. Acesso em: 08 dez 2019. Citado na página 25.

TONIETTO, L.; WALTER, M. Análise de algoritmos para chroma-key. *Monografia para obtenção de grau de Bacharel em Informática. São Leopoldo*, 2000. Citado 2 vezes nas páginas 7 e 18.

TORI, R.; KIRNER, C.; SISCOUTTO, R. A. *Fundamentos e tecnologia de realidade virtual e aumentada*. [S.l.]: Editora SBC, 2006. Citado na página 14.

VUFORIA.COM. *Supported Versions*. https://library.vuforia.com/articles/Solution/Vuforia-Supported-Versions: [s.n.], 2019. Citado na página 29.