CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 3.607, de 17/10/05, D.O.U. nº 202, de 20/10/2005
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Thayso Henrique Capuchinho Camargo

AVALIAÇÃO DA AMPLITUDE DE MOVIMENTO (ADM) EM PLANO CORONAL ANTERIOR UTILIZANDO O SENSOR KINECT: articulações do ombro e do quadril

Palmas - TO
2015

Thayso Henrique Capuchinho Camargo

AVALIAÇÃO DA AMPLITUDE DE MOVIMENTO (ADM) EM PLANO CORONAL ANTERIOR UTILIZANDO O SENSOR KINECT: articulações do ombro e do quadril

Trabalho de Conclusão de Curso (TCC) elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Mestre Fernando Luiz de Oliveira.

Palmas - TO

2015

Thayso Henrique Capuchinho Camargo

AVALIAÇÃO DA AMPLITUDE DE MOVIMENTO (ADM) EM PLANO CORONAL ANTERIOR UTILIZANDO O SENSOR KINECT: articulações do ombro e do quadril

Trabalho de Conclusão de Curso (TCC) elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Mestre Fernando Luiz de Oliveira.

Aprovado em: ____/____/_____

BANCA EXAMINADORA

_____

Prof. M.Sc Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Fabiano Fagundes

Centro Universitário Luterano de Palmas

_____

Prof. M.Sc. Pierre Soares Brandão

Centro Universitário Luterano de Palmas

Palmas - TO

2015

RESUMO

Os profissionais fisioterapeutas utilizam técnicas como goniometria e fleximetria na captura de ângulos corporais de Amplitude de Movimento (ADM) em avaliações e tratamentos fisioterapêuticos. Porém, para a aplicação dessas técnicas é necessário ter habilidade para manusear a ferramenta. Tal habilidade que pode variar de acordo com a experiência e conhecimento do profissional, e assim, consequentemente apresentar uma variabilidade tanto inter-avaliador como intra-avaliador nos valores capturados durante o processo de avaliação. Diante de tal situação, o presente projeto trata a respeito do desenvolvimento de uma ferramenta que utiliza o sensor *Microsoft Kinect for Windows* juntamente com o *Kinect for Windows SDK* na realização da captura dos ângulos corporais de ADM, com o objetivo de diminuir ou eliminar o fator humano nos valores capturados durante o processo de obtenção dos ângulos corporais.

Palavras-chave: Amplitude de Movimento, Sensor Microsoft kinect, Ferramenta Fisioterapêutica.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADM	Amplitude de Movimento
ADMP	Amplitude de Movimento Passivo
ADMA	Amplitude de Movimento Ativo
ADMA-A	Amplitude de Movimento Ativo-Assistido
API	Applications Programming Interface
CEULP	Centro Universitário Luterano de Palmas
CMOS	Complementary Metal Oxide Semiconductor
ETM	Erro Técnico de Medida
IDE	Integrated Development Environment
IR	Infra Red (infravermelho)
RGB	Padrão de cores de imagens (Vermelho, Verde, Azul)
SDK	Software Development Kit
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language

SUMÁRIO

1 INTRODUÇÃO

A tecnologia vem a cada dia ocupando um espaço maior na realização de atividades e facilitando na execução de diferentes tarefas, que vão desde um simples cálculo matemático até a realização de cirurgias complexas. Isto é verificado pelas sucessivas pesquisas e ferramentas que surgem a cada dia, nas quais acabam permitindo ampliar ainda mais as suas áreas de atuação.

A saúde é uma das áreas que tem procurado informatizar suas atividades, fazendo com que a tecnologia tenha um papel importante na realização de diagnósticos, cirurgias, entre outros. Porém, ainda existem muitas atividades que são desenvolvidas de forma manual e/ou por meio de ferramentas manuais. Um exemplo disto consiste no trabalho do fisioterapeuta na obtenção das medidas antropométricas, de movimentos articulares, entre outras que são usadas como referência na avaliação e tratamento de pacientes.

Os fisioterapeutas capturam os ângulos corporais através de técnicas como a goniometria e fleximetria, nas quais utilizam ferramentas como o flexímetro e o goniômetro, aparelhos antigos que necessitam de uma habilidade na forma de manipula-los. Silva et al. (2011) afirma que técnicas como a goniometria e fleximetria apresentam uma baixa confiabilidade ao serem utilizadas na reavaliação de medidas.

No entanto, como dito anteriormente, a tecnologia pode e deve ser utilizada para facilitar o trabalho dos profissionais envolvidos em determinadas tarefas. Por exemplo, o Kinect, por ser um sensor que utiliza técnicas da visão computacional para a captura de imagens, identificação de objetos, rastreamento e mapeamento de pontos do corpo humano, se torna um possível método de captura de ângulos corporais. A sua utilização permitiria obter a Amplitude de Movimento (ADM) de pacientes em seções fisioterapêuticas.

Este projeto teve por objetivo propor, desenvolver e testar uma ferramenta confiável que utilizasse o sensor *Microsoft Kinect for Windows* na realização da captura dos ângulos corporais de ADM, encontrados nas articulações dos ombros e do quadril, realizado em movimentos ativos e encontrado em um plano coronal de visão anterior. Esta tarefa se tornou possível através da utilização de cálculos e técnicas matemáticas aplicadas sobre os pontos das articulações, nos quais são formados pelo rastreamento e mapeamento do esqueleto realizado pelo software de desenvolvimento *Kinect for Windows SDK*.

Este trabalho encontra-se estruturado em seções, nas quais apresentam todos os passos percorridos para se chegar à ferramenta implementada, sendo que na seção 2 é apresentada a revisão de literatura, um referencial teórico resultante de pesquisas bibliográficas contendo os

principais conceitos envolvidos no projeto; em 3, são apresentados os materiais utilizados no desenvolvimento da ferramenta, além da metodologia utilizada no trabalho; em seguida, seção 4, são exibidos os resultados obtidos. As considerações finais alcançadas ao término do projeto são discutidas em 5, tendo ao final a seção 6, que apresenta as referências bibliográficas utilizadas durante o trabalho.

2 REFERENCIAL TEÓRICO

Nesta seção serão abordados conceitos importantes relacionados ao trabalho, tais como: a Amplitude de Movimento e suas técnicas, Métodos de Avaliação Corporal, Análise de movimentos e ferramentas necessárias para à captura de dados e desenvolvimento da aplicação.

2.1 Amplitude de Movimento (ADM)

ADM pode ser definido como o movimento completo possível de ser realizado por cada articulação. Estes movimentos acabam envolvendo músculos, fáscias, cápsula, ligamentos, tendões, vasos, nervos; e podem ser trabalhados de diferentes formas, de acordo com a necessidade do paciente e a escolha do profissional fisioterapeuta (KISNER; COLBY, 1998). É válido ressaltar que a ADM é composta pela amplitude de movimento articular na qual se refere o movimento presentes em uma articulação, e amplitude de movimento muscular, sendo o movimento articular proveniente do comprimento dos músculos. Ambas as amplitudes são expressas em termos de grau (KENDALL; MCCREARY; PROVANCE, 1995), o qual acaba sendo um ponto de referência para o fisioterapeuta ao informa a medida durante a realização do movimento da articulação. Isto porque a “medida da Amplitude de Movimento (ADM) articular é um relevante parâmetro na avaliação física, permitindo aos profissionais acompanhar de modo quantitativo a evolução do tratamento” (BATISTA; MEIRA; SANTANA, 2010, p.85).

Para que seja identificada a amplitude completa de uma articulação são utilizadas ferramentas de medida angular como o goniômetro “instrumento mais utilizado para medir a amplitude de movimento” (MARQUES, 2003, p. 1), que aplicado sobre um conjunto de movimentos pode identificar o ângulo da articulação, servindo de parâmetro para que o fisioterapeuta informe se a mobilidade se encontra normal ou não.

Diversos fatores acabam contribuindo para que haja uma redução na mobilidade, como doenças articulares, sistemáticas, neurológicas e musculares, nas quais podem ser causadas por agressões cirúrgicas, imobilidade ou inatividade, acarretando em uma diminuição na ADM da articulação. Isto é ocasionado porque quando uma pessoa executa as atividades diárias os tecidos e articulações se alongam e encurtam-se continuamente, mantendo assim sua mobilidade. Porém, se a movimentação normal for restringida, irá acarretar em um encurtamento adaptativo (KISNER; COLBY, 1998).

> O encurtamento adaptativo progressivo ocorre quando o corpo responde a uma redução de carga. Esse encurtamento limita a mobilidade e a função, reduzindo a capacidade de o paciente realizar as atividades normais da vida diária, no trabalho ou nas atividades de lazer. O paciente se adapta a essas limitações, recorrendo a outras articulações ou membros para alcançar os objetivos funcionais, contribuindo assim para o desuso (HALL; BRODY, 2001, p. 89).

Então, para que se possa manter uma mobilidade normal é necessário que haja um comprimento adequado dos tecidos, movimentando os segmentos em suas amplitudes completas para permitir uma ADM plena e uma habilidade neuromuscular para execução desses movimentos. Essa ativação dos membros e articulações acaba sendo mantida na maioria dos indivíduos devido a sua utilização diária (HALL; BRODY, 2001).

Para Kisner e Colby (2009) a ADM pode ser considerada uma técnica básica que utiliza um conjunto de exercícios que visam avaliar e contribuir na normalização dos movimentos necessários para realizar atividades funcionais. Isto ocorre por causa da aplicação de forças musculares ou auxiliares que permitem movimentar os ossos em diferentes padrões ou ADM's.

Dentre os três padrões existentes na ADM, encontram-se a Amplitude de Movimento Passivo (ADMP), Amplitude de Movimento Ativo-Assistido (ADMA-A), e a Amplitude de Movimento Ativo (ADMA). Na primeira, os movimentos são executados nas articulações utilizando totalmente uma força externa, sem contração muscular voluntária. Este tipo de mensuração é normalmente indicada para pacientes que não possui capacidade ou são proibidos de realizar de forma voluntária/ativa, ou seja, o movimento é realizado por outra pessoa, por um aparelho, pela gravidade ou de forma mecânica.

> A ADM passiva é usada quando o movimento ativo pode romper o processo de cicatrização, quando o paciente é físico ou cognitivamente incapaz de se movimentar ativamente, ou quando a realização do movimento ativo é por demais dolorosa. Os movimentos passivos são usados, também, para ensinar os exercícios ativos ou de resistência e para reproduzir relaxamento (HALL; BRODY, 2001, p. 96).

Os exercícios de AMDP acabam sendo aplicados com a finalidade de se manter uma integridade da articulação ou tecido mole; minimizar efeitos da formação de contraturas; manter uma elasticidade mecânica dos músculos, assistir a circulação e dinâmica vascular; melhorar os movimentos sinoviais para nutrição das cartilagens e difusão de substâncias dentro da articulação; diminuir ou inibir a dor; auxiliar o processo de cicatrização após uma lesão ou cirurgia e ajudar a manter a consciência de movimento no paciente (KISNER; COLBY, 1998).

Outro padrão existente é a Amplitude de Movimento Ativo-Assistido (ADMA-A) no qual possui semelhanças com a ADMP, porém o movimento aplicado nesta ADM é executado

pela movimentação ativa das articulações, contando com a assistência de forças externas, como uma contribuição de uma força aplicada pelo fisioterapeuta, gravidade, mecânica ou do próprio paciente utilizando outro segmento do corpo. Essa ADM é utilizada para pacientes que possuem uma musculatura fraca, geralmente sendo incapazes de ativar plenamente os músculos ou apenas não possuem permissão. Normalmente esse padrão é indicado quando o paciente já está apto ou se deseja alguma ativação muscular (HALL; BRODY, 2001), pois a realização deste tipo de exercício serve para alcançar as mesmas metas dos exercícios de ADMP, só que possuem um benefício extra por realizarem contrações musculares.

Por fim, o terceiro padrão refere-se à Amplitude de Movimento Ativa (ADMA), na qual o exercício aplicado é produzido pela contração dos músculos e sem a necessidade de uma contribuição externa.

> Refere-se à quantidade de movimento articular realizado por um indivíduo sem qualquer auxílio. Quando a amplitude é realizada ativamente, o examinador tem a informação exata sobre a capacidade, coordenação e força muscular da amplitude de movimento do indivíduo (MARQUES, 2003, p. 5).

A ADMA é um excelente movimento para se obter a ADM de uma articulação, porém a limitação da mobilidade ativa pode ser causada pelos mesmos motivos que a mobilidade passiva, como encurtamento, rigidez, espasmo, contratura entre outros fatores que impõem um limite na capacidade da articulação movimentar-se através da ADM (HALL; BRODY, 2001).

Para Kisner e Colby (1998, p.27) a ADMA deve ser aplicada “Quando o paciente está apto para contrair ativamente seus músculos e mover um segmento com ou sem assistência, e quando não existem contra-indicações [...] Quando um paciente é colocado em um programa de condicionamento aeróbico”. Essa ADM tem por finalidade alcançar as mesmas metas da ADMA-A, assim como manter a elasticidade e contratibilidade dos músculos; servir como um *feedback* sensorial dos músculos contraídos; prover estímulos para uma integridade óssea e articular; aumentar a circulação para prevenir a formação de trombos; e desenvolver coordenação e habilidades motoras.

Algumas precauções devem ser tomadas durante todo o processo de realização dos exercícios de ADM para que se possa manter a integridade do paciente, sendo necessário respeitar a tolerância de cada indivíduo para que o mesmo não sofra com dores excessivas.

Para Kisner e Colby (1998) o fisioterapeuta deve ficar em alerta durante todo o processo de avaliação, sendo aconselhável para que o mesmo sempre verifique as condições do paciente, como a observação dos sinais vitais notando se há alguma alteração na coloração do segmento ou na amplitude do movimento realizado. Estas reações se tornam um

importante parâmetro ao avaliar a procedência do tratamento, se é possível progredir ou necessário regredir.

O fisioterapeuta sempre deve acompanhar de perto o paciente durante a execução dos exercícios, pois não se pode realizar movimentos excessivos ou errados, para não haver um aumento da dor ou inflamação. É importante ressaltar que qualquer exercício que possa trazer prejuízo à saúde do paciente se torna contraindicado, como situações em que o movimento será realizado após uma ruptura aguda; ter sofrido fraturas e cirurgias; ter sofrido derrames articulares ou hemorragias; após infarto ou miocárdio; possua neoplasia ou algum trauma adicional.

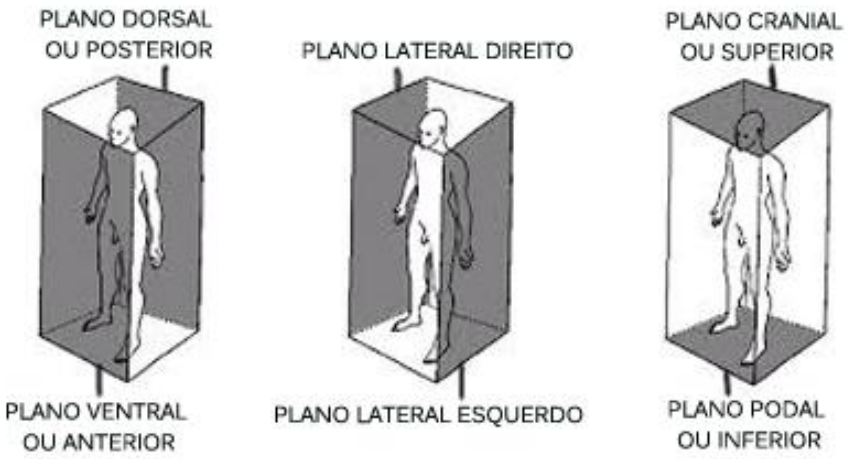
2.2 Planos de Delimitação do Corpo Humano

O corpo humano em sua posição anatômica possui diferentes planos imaginários que delimitam e tangenciam a superfície corpórea, dos quais demarcam o corpo ou parte deste. Nobeschi (2010) apresenta seis planos de delimitação do corpo humano, os quais são apresentados a seguir:

- **Plano ventral ou anterior** - tangente à parte anterior do corpo.
- **Plano dorsal ou posterior** - tangente à parte posterior do corpo.
- **Plano lateral direito** - tangente à parte lateral direita do corpo.
- **Plano lateral esquerdo** - tangentes à parte lateral esquerda.
- **Plano cranial ou superior** - tangente à parte superior do corpo.
- **Plano podal ou inferior** - tangente à parte inferior do corpo.

Para facilitar o entendimento, a figura 1, a seguir, apresenta estes planos tendo como referência o corpo humano.

Figura 1 - Planos de Delimitação corporal de um ser humano.



Fonte: Adaptação de Marlon Saldanha (2014, online).

2.3 Planos de Secção e os Eixos Anatômicos

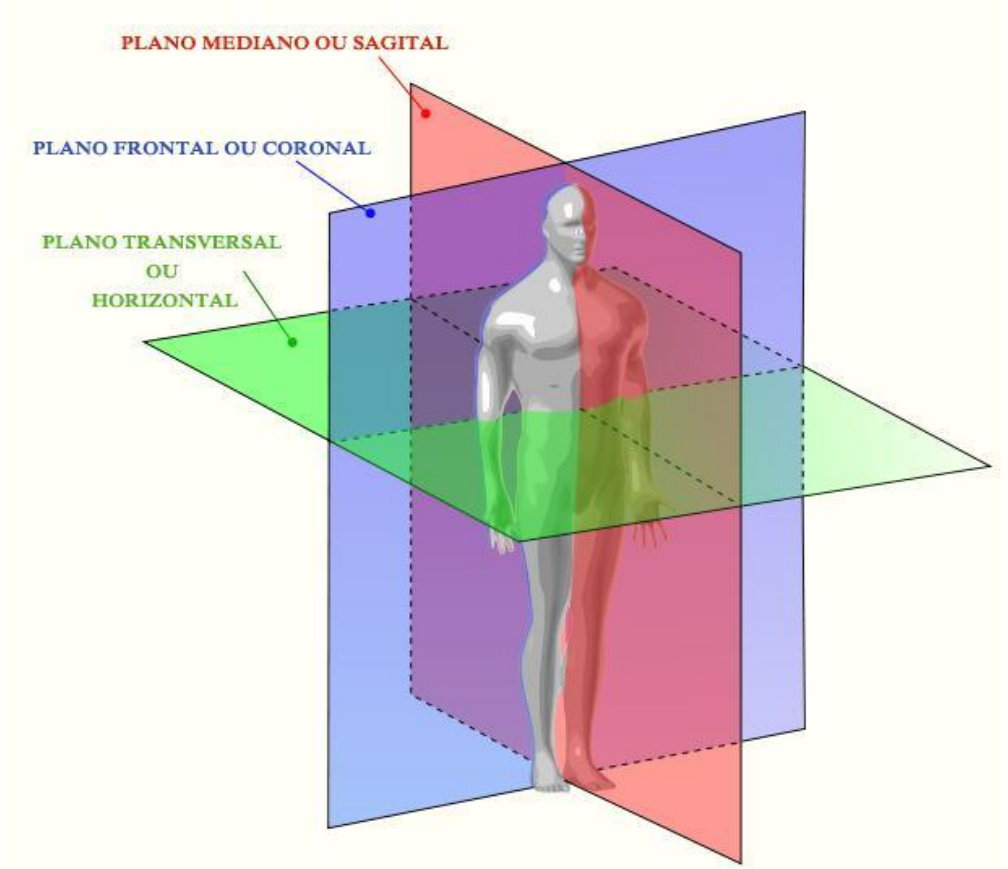
Os planos de secção anatômica são planos que cortam ou orientam o corte do corpo humano, sendo definido por Kendall, McCreary e Provance (1995) como três planos principais para referência, sendo os planos sagital, coronal e transversal. Porém Moore e Dalley (1999) afirma que as descrições anatômicas são baseadas em quatro principais planos imaginários, pois considera básica a divisão do plano sagital em mediano e paramediano.

Estes planos podem ser proferidos em diferentes formas como pode ser descrito a seguir:

- **Plano Mediano ou Sagital:** plano que vai da face anterior à face posterior do corpo e divide o corpo em duas metades: direita e esquerda. Neste plano ocorrem os movimentos de flexão e extensão (MARQUES, 2003). O plano sagital também pode ser caracterizado de duas formas:
  - **Plano sagital mediano:** onde o plano divide as duas metades em partes semelhantes.
  - **Plano sagital paramediano**: onde o plano divide as duas metades em partes distintas.
- **Plano Frontal ou Coronal:** o plano que vai de um lado ao outro do corpo, dividindo-o em duas metades, a parte da frente e a parte de trás. Os movimentos que se encontram neste plano são os de adução e abdução (MARQUES, 2003).
- **Plano Transversal ou Horizontal:** o plano horizontal que divide o corpo em partes superior e inferior. Neste plano ocorrem os movimentos de rotação (MARQUES, 2003).

Os planos anatômicos podem ser ilustrados por meio da figura 2, apresenta a seguir.

Figura 2 - Planos Anatômicos do corpo humano.



Fonte: Adaptação de Nick Zuccarello (2014, online).

Kendall, McCreary e Provance (1995, p.11) definem os eixos como “linhas, reais ou imaginárias, em torno das quais ocorrem os movimentos” e possuem como referência os planos sagital, coronal e transversal.

2.4 Movimentos Corporais

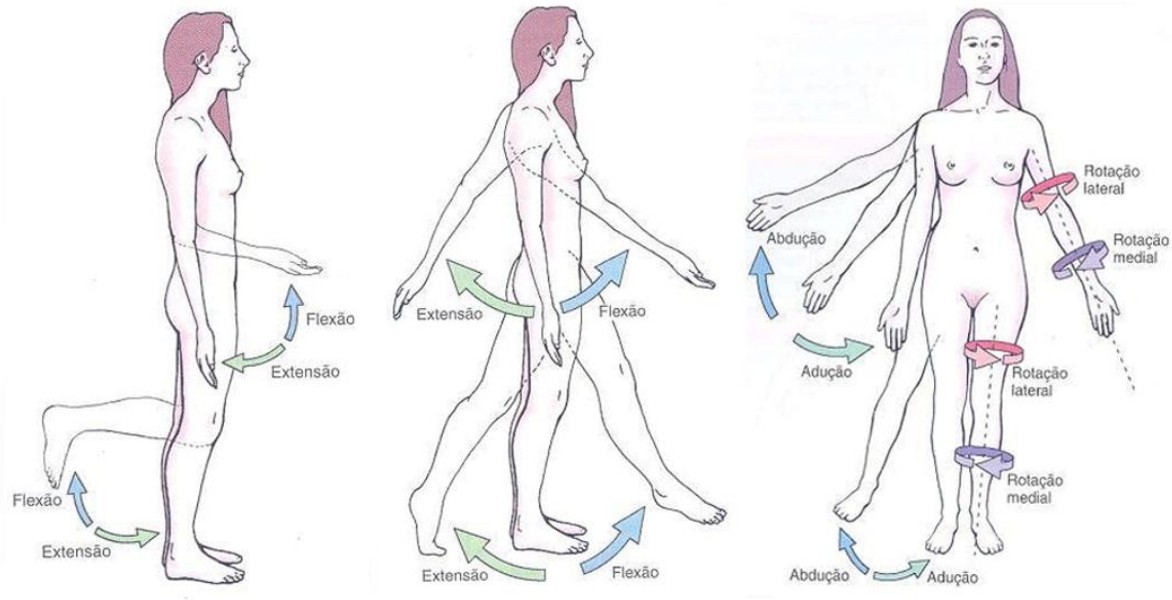
O movimento articular pode ser definido, conforme Correia (2012, p. 69), como “a variação de movimento de um corpo rígido rodando em torno de um eixo”, onde o eixo corresponde a articulação e serve como um ponto de referência para o cruzamento dos segmentos. Os tipos de movimentos existentes se dividem em termos de flexão, extensão, abdução, adução e rotação, nas quais ocorrem nos diferentes eixos apresentados na seção anterior.

Nosso corpo realiza movimentos diários para poder manter a mobilidade normal. Essa mobilidade é definida por Hall e Brody (2001) como “a presença de uma amplitude funcional através da qual é possível movimentar-se e a capacidade de iniciar e manter um movimento ativo através da amplitude”. Segundo Neumann (2011) durante os movimentos ocorre interações mecânicas e fisiológicas entre os músculos e as articulações, nas quais permitem a

realização de movimentos comuns durante a realização de atividade como, caminhar, correr, dançar e pular.

Para realização destas atividades executa-se os movimentos existentes nos termos citados anteriormente, nos quais se encontra os movimentos que podem ser identificados pela figura 3.

Figura 3 - Tipos de Movimentos Angulares.



Fonte: Adaptação de www.auladeanatomia.com (2014, online).

A figura 3 apresenta os termos de Flexão, Extensão, Abdução, Adução e Rotação, onde a Flexão é realizada pela curvatura ou diminuição dos ângulos das articulações, sendo "o movimento na direção anterior para a cabeça, pescoço, tronco, membros superiores e quadril" (KENDALL; MCCREARY; PROVANCE, 1995, p.13).

O movimento de Extensão é executado de forma que realize o aumento do ângulo entre as articulações, sendo o movimento realizado no sentido oposto à flexão. Onde ambos os movimentos de extensão como os de flexão são realizados em um plano sagital.

Na Abdução os movimentos são realizados em direção contrária ao plano sagital, tendendo a se afastar do centro do corpo, diferente dos movimentos de Adução nos quais os movimentos se aproximam do plano sagital, sendo estes movimentos realizados em um plano coronal.

Existem também os movimentos de rotação, que para Kendall, McCreary e Provance (1995, p.14) "refere-se ao movimento ao redor de um eixo longitudinal, no plano transverso para todas as áreas do corpo, exceto escápula e clavícula". O movimento de rotação se divide

em Rotação Medial e Rotação Lateral, sendo o primeiro uma rotação transversa orientada para a parte anterior do corpo, na qual ocorre quando a parte anterior do segmento se movimenta para o plano mediano do corpo, ou seja, o braço realiza o movimento de rotação para dentro. Já a Rotação Lateral é uma rotação transversa orientada para a superfície posterior do corpo, onde a parte anterior do segmento se movimenta para o plano lateral do corpo, sendo o movimento de rotação que rotacional o braço para fora.

2.5 Variabilidade Intra-Avaliador e Inter-Avaliadores

A utilização de medidas é a principal forma de se obter as informações durante as avaliações, sendo usadas tanto para situações simples como complexas, com a finalidade de se obter dados que exponham confiança, ou seja, dados com a maior precisão possível (MATHEWS, 1980). Porém, toda medida possui erro, nos quais não podem ser quantificados com exatidão, pois ninguém sabe o verdadeiro resultado de uma medida, apenas se descreve uma zona de valores na qual esperamos que contenha o valor verdadeiro (KISS, 2003).

Conforme Pereira (1995) os erros podem ser classificados em três categorias, sendo essas, as situações onde são realizadas as medidas, os instrumentos usados para mensuração, ou as pessoas envolvidas no processo nas quais se encontram o observado ou o observador. Esse conjunto de erros acaba por interferir no resultado das medidas, causando assim uma variação.

Essa variabilidade dos resultados ainda é um grande problema enfrentado por profissionais fisioterapeutas durante o processo de mensuração das medidas corporais, tanto antropométricas como de ADM. Esse problema pode ser identificado devido a aplicação dos cálculos do Erro Técnico de Medida (ETM).

> O cálculo do ETM fornece indicadores de precisão e representa o controle de qualidade da medida, podendo ser realizado para estimar a variabilidade intra-avaliador - variabilidade das medidas realizadas pelo mesmo avaliador em dias diferentes, no mesmo avaliado ou grupo de avaliados, e inter-avaliadores - comparação da variação das medidas realizadas por dois ou mais avaliadores em um mesmo grupo de avaliados (FILHO et al., 2007, p2).

Para Castro et al (2008) a variabilidade das medidas pode ser decorrente de variação biológica ou de inadequação técnica na obtenção dos valores, onde a maior parte do ETM é passível de controle, por meio da calibração dos entrevistadores. Porém, o resultado da pesquisa de avaliação de medidas antropométricas intra-avaliador e inter-avaliadores realizada por Filho et al. (2007), mostra que mesmo com uma padronização recomendada para a realização das medidas, houve um alto nível de variabilidade. Neste estudo foi constatado que até mesmo profissional com tempo de experiência no campo, com cursos teóricos e práticos, teria a necessidade de realizar um aprimoramento técnico, pois os resultados apresentaram

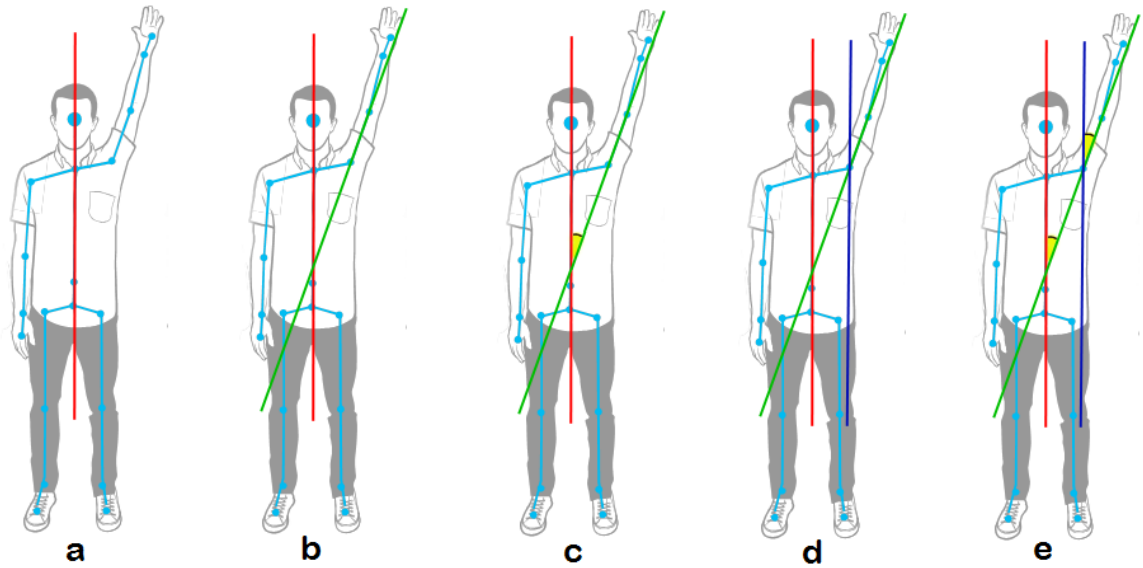
que mais de 50% das medidas continham ETM’s acima dos padrões recomendados, tanto para as avaliações intra-avaliador como inter-avaliadores.

Resultados de variabilidade não se aplicam apenas nas medidas antropométricas, pois podem ser encontradas variâncias durante a mensuração de ângulos, mesmo que os valores capturados sejam aceitáveis. Isto foi verificado no resultado de estudos aplicados por Santos et al. (2011) sobre a confiabilidade de medidas angulares. Neste estudo, mesmo com a utilização de técnicas confiáveis como a fotogrametria e a goniometria para a captura da ADM, apontam a detecção de uma variação nas medidas capturadas, no qual o autor da pesquisa ressalta que os dados capturados devem ser analisados com cautela para não comprometer a eficácia de tratamentos.

2.6 Mensuração de Movimentos Articulares (ângulos)

A ADM é a quantidade de movimento possível de ser realizado por uma articulação, que pode ser expressa em grau o valor do ângulo alcançado pelo movimento. Machado (1986) caracteriza um ângulo por “um par de semi-retas de origem no mesmo ponto”. Então para encontrar o ângulo em um exercício de ADM é necessário obter os dados relacionados aos segmentos do corpo e à articulação, e aplicar essas informações sobre técnicas e cálculos matemáticos para assim encontrar o ângulo da articulação. Os meios utilizados para captura do ângulo são descritos a seguir (Figura 4).

Figura 4 - Método de obtenção do ângulo.



Fonte: Adaptação de Microsoft (2014, online).

Os dados devem ser obtidos sobre um plano anterior do corpo do paciente, sendo necessário o rastreamento e mapeamento dos pontos articulares, montando assim um esqueleto do paciente. A partir daí, será criada a matriz determinante da equação geral da reta (Figura 5) por meio dos pontos do tórax e do abdômen como descrito na figura 4a, e os pontos do ombro e cotovelo como descrito na figura 4b.

Figura 5 - Matriz Determinante da Equação Geral da Reta.

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x & y & 1 \end{vmatrix} = 0$$

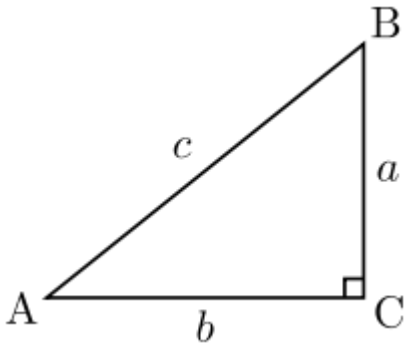
Fonte: www.alunosonline.com.br (2014, online).

Tendo esta matriz, será possível identificar a equação geral de cada uma das retas (1). Giovanni e Bonjorno (2005) fala que toda reta possui uma equação geral, assim é possível estender as duas retas e identificar o ponto de intersecção das mesmas.

$$ax + by + c = 0 \qquad (1)$$

Após a identificação das retas e do ponto de intersecção entre as mesmas, tem-se os valores x e y dos pontos A, B e C do triângulo, como apresentado na Figura 6.

Figura 6 - Triângulo de representação da lei dos cossenos.



Fonte: Machado (1998, p.25).

Estes valores aplicados sobre (2), permitirá encontrar a distância entre os pontos A, B e C, sendo os valores a, b e c do triângulo (Figura 6).

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (2)$$

Após encontrar todos os pontos, ter-se-á um triângulo que pode ser comparado ao gerado pelo movimento e cruzamento das retas, como apresentado na figura 4b. A partir deste ponto, é possível a descoberta de qualquer um dos ângulos, através da Lei dos Cossenos (3), que diz que "em todo triângulo, o quadrado de um lado é igual à soma dos quadrados dos outros dois lados, menos duas vezes o produto desses lados pelo cosseno do ângulo que eles formam" (MACHADO, 1998, p.25). A lei dos Cossenos pode ser aplicada a qualquer um dos lados do triângulo, permitindo que seja encontrado o ângulo apresentado na figura 4.C.

$$\begin{aligned} a^2 &= b^2 + c^2 - 2.b.c.\cos\theta \\ b^2 &= a^2 + c^2 - 2.a.c.\cos\beta \\ c^2 &= a^2 + b^2 - 2.a.b.\cos\alpha \end{aligned} \qquad (3)$$

Pode-se notar na figura 4d que a reta vertical utilizada para realização do cálculo não se encontra ligada à articulação na qual o movimento é executado, pois o sensor Kinect não possui um ponto inferior que possa identificar uma reta válida. Porém a criação de uma reta vertical paralela ao eixo do movimento permitirá que seja encontrado o ângulo desejado, pois as retas paralelas possuem o mesmo coeficiente angular como pode ser visto na figura 4e.

2.7 Visão Computacional

Para o ser humano a visão ainda é o principal meio de identificação de ambientes, no qual é descrito por Mattos (2012) como o principal elemento sensorial para sobrevivência dos seres vivos, por fornecer a maior quantidade de informações. Estas informações são essenciais para identificação de alimento, obstáculos ou predadores.

Por este motivo tem-se criado e aprimorado aplicações que proporcionem às máquinas adotarem funções do sistema de visão dos seres vivos, sendo assim criada a área na qual é chamada de visão computacional.

Segundo Ballard e Brown (1982), a visão computacional é a ciência responsável pela visão de máquinas, dos meios e tecnologias que permitem um computador enxergar o ambiente à sua volta, extraindo informações de imagens ou outros dados multidimensionais, dos quais podem ser capturados por meio de câmeras de vídeo, sensores, scanners, entre outros dispositivos. A partir dessas informações permite-se reconhecer e manipular os objetos existentes na imagem.

A área da visão computacional se torna um ponto forte para o desenvolvimento de aplicações relacionadas à diferentes áreas, que vão desde a medicina, na realização de diagnósticos de doenças, segurança na realização de reconhecimento biométrico, às áreas

industriais, permitindo uma automatização de robôs e máquinas para auxiliarem em tarefas, como a de inspeção industrial.

Para realização destas tarefas são empregados diferentes campos da visão computacional, nas quais se caracterizam pela utilização de imagens e técnicas de reconhecimento de padrões, processamento de imagens, fotogrametria entre outras (NETO; BRUNO, 2012).

Segundo Tavares (2000) o processamento de imagens e a visão computacional são divididos em quatro áreas de atuação, atuando no melhoramento ou realce de imagens, restauração de imagens, compressão de imagens e análise de imagens. As três primeiras costumam ser utilizadas no processamento de imagens, para melhoramento e realce de características da imagem, restauração de imagens que tenham sido degradadas, e representar a imagem de uma forma mais simples, deixando a imagem mais leve, mas sem fazer com que haja a perca as informações necessárias. Já a última se torna mais voltada a visão computacional, pois consiste na interpretação de uma ou mais imagens, na tentativa de se medir, reconhecer e classificar as mesmas. Sendo muitas vezes associada à inteligência artificial para realização destas tarefas.

As técnicas de visão computacional não são aplicadas apenas em imagens estáticas, mas também permitem uma análise de movimentos por meio de uma sequência de imagens 2D ou 3D. Segundo Mattos (2012) a visão computacional possibilita a análise de movimentos humanos, nos quais permitem o desenvolvimento de sistemas na área médica e fisioterapêutica.

2.8 Kinect

O sensor de movimento *Microsoft Kinect* é um dispositivo que fornece uma visão computacional sobre ambientes, no qual permite uma captura de informações através de seus sensores. Inicialmente possuía o codinome de Projeto Natal, no qual o nome fazia referência a capital do estado do Rio Grande do Norte.

Encabeçado pela *Microsoft* em colaboração com a empresa israelita *Primesense,* possuía como seu idealizador o brasileiro Alex Kipman. Esse dispositivo inicialmente visava criar uma nova tecnologia capaz de permitir aos jogadores interagir com os jogos eletrônicos sem a necessidade de ter em mãos um controle/*joystick*.

O que inicialmente seria apenas utilizado com o propósito de diversão com games, se mostrou ser um aparelho com várias funções e pode ser utilizado por inúmeras aplicações desenvolvidas para seu console de videogame Xbox 360, com a finalidade de competir com o *Wii Remote Plus* da Nitendo e o *PlayStation Move* da Sony, sendo capaz de reconhecer o

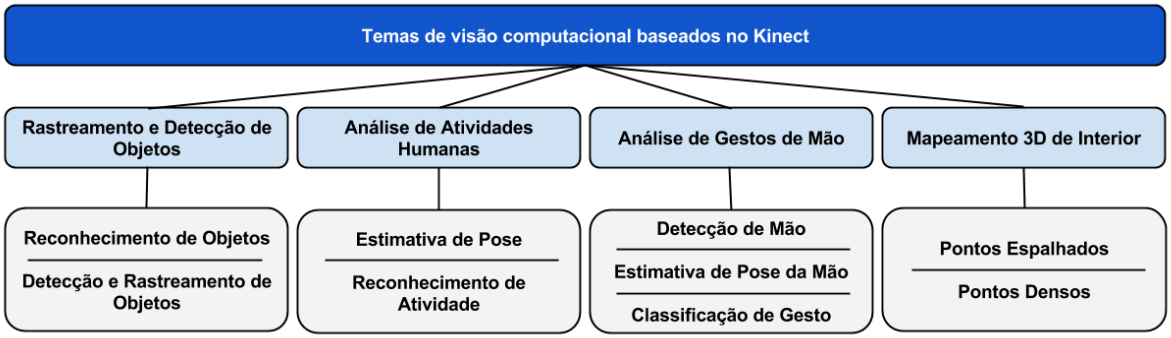
movimento de pontos do corpo humano, além de face e voz, através de sensores especiais (ZANDONA; TANAKA, 2012).

Por possuir uma grande capacidade de captura e interação com o jogador o sensor acabou despertando uma curiosidade e segundo Silveira (2011) o sucesso do Kinect foi tão grande que a *Microsoft* resolveu investir em outros usos, permitindo que os seus recursos pudessem ser manipulados por meio de *softwares* como o *Kinect Software Development* (SDK).

O Kinect por ser um dispositivo de visão computacional que possui coordenadas com um marcador de erro menor que 1 centímetro na detecção de objetos, se torna apto para a realização de inúmeras tarefas.

Para Yamada et al. (2012) a visão computacional ainda tem sofrido bastante para realizar o mapeamento de superfícies e a geração de modelos de objetos. Porém, Jungong et al. (2013) entende o sensor Kinect como um dispositivo capaz de resolver diferentes tipos de problemas enfrentados pela visão computacional, dos quais se encontram o rastreamento e reconhecimento de objetos, análise de atividades humanas, análise de gestos de mão e mapeamento tridimensional de interiores, como pode ser visualizado pela Figura 7.

Figura 7 - Visão Computacional melhorada pelo sensor Kinect.



Fonte: Adaptação de Jungong et al. (2013, p. 1319).

Como observado, a resolução de problemas enfrentados pela visão computacional pode ser facilitado através da utilização do sensor Kinect, no qual permite:

- Rastreamento e Detecção de objetos: se divide em detectar, rastrear e em reconhecer um objeto. Primeiramente, a partir de imagens RGB capturadas por câmeras e sensores, nas quais se aplicam algoritmos para a remoção do fundo das imagens, assim permitindo a detecção de objetos. Após encontrar os objetos, utiliza-se, cores, movimentos e a textura das imagens RGB na aplicação de algoritmos, nos quais permitem o reconhecimento dos objetos.

- Análise de atividades humanas: tem por objetivo alcançar com velocidade e precisão a postura das articulações ósseas e analisar os movimentos realizados por seres humanos. O sensor Kinect inova ao fornecer um avançado rastreamento de esqueleto, no qual permite o mapeamento de pontos do corpo humano. Permitindo assim que se crie uma estimativa de poses, e possibilitando a aplicação de métodos para reconhecimento de movimentos.
- Análise de gestos de mão: com a finalidade de permitir uma melhor interação com a máquina a análise de gestos se divide na, detecção da mão, estimativa de pose da mão e classificação de gestos. A detecção da mão se assemelha à detecção de objeto, utilizando as características da imagem RGB e de profundidade da mão. Na estimativa de pose se baseia em modelos 3D e em um conjunto pré-definido de características de formato das mãos, nas quais permitem uma estimativa das poses realizadas. Por fim, a classificação dos gestos geralmente incorpora a detecção de mão e representar técnicas de saída, baseadas em palavras-chave e estabelecimento de identificação de gestos simples como agarrar e apontar.

Mapeamento 3D de interiores: busca o mapeamento e uma representação digital de ambientes internos, nos quais utilizam os pontos densos, os quais monitoram dois quadros e utilizam todos os pixels densos para o alinhamento da cena, e os pontos de esparso, no qual baseia-se na reconstrução de pontos coincidentes e distintos, extraídos de sucessivos quadros, para identificação de formas geométricas.

2.8.1 Arquitetura

A arquitetura do sensor Kinect é composta por diferentes componentes que realiza a identificação do ambiente a sua frente. Os principais componentes são: uma câmera RGB que permite o reconhecimento corporal das pessoas que se encontram à frente do console; um projetor que produz feixes de luz infravermelho; um sensor de profundidade que permitem o mapeamento em três dimensões do ambiente à frente; quatro microfones embutidos que permite captar e identificar vozes de diferentes pessoas no ambiente, além de identificar em qual direção o som está vindo e cancelar ruídos e ecos que possam atrapalhar na identificação das vozes (QUEIROZ, 2010).

O sensor também possui um motor que permite a inclinação do mesmo. Esses componentes podem ser melhor visualizados na Figura 8, a qual apresenta o sensor Kinect e seus principais componentes como as suas localizações.

Figura 8 - Sensor do Kinect.

Fonte: www.samuelr.com (2014, online).

A câmera RGB fica localizada ao centro do dispositivo e permite uma resolução de até 1280x1024 em 15 imagens por segundo, mas possui o uso habitual de 640x480 em 30 imagens por segundo à 32-bit cor, ou seja, a cada frame é gerada uma nuvem de pontos que possui 307200 pontos. Para cada ponto mapeado, são fornecidas informações de profundidade e de cor RGB (MATTOS, 2012; COUTO, 2012).

Logo a esquerda do dispositivo possui um projetor de luz infravermelho chamado de "IR *light*" que utiliza a técnica de luz projetada, no qual lança uma matriz de pontos infravermelhos, os quais permitem com que a câmera de profundidade CMOS que fica localizada à direita do dispositivo possa capturar esses pontos e calcular um ambiente em 3D (MICROSOFT, 2014).

Para a entrada de áudio o dispositivo possui 2 microfones em cada lado que permitem o reconhecimento de voz, cancelamento de eco e identificação do ponto onde está sendo gerado o som no ambiente. Possui também uma base motorizada que permite a inclinação do sensor em aproximadamente 27º para cima ou para baixo (BRUNNER, 2012).

Essas especificações relacionadas aos sensores encontrados no Kinect pode ser resumida pela tabela apresentada a seguir.

Tabela 1 - Arranjo de especificações dos sensores.

Sensor Item	Faixa de especificações
Ângulo de visão	Campo de visão de 43º vertical por 57º horizontal
Ranger de inclinação mecanizado (vertical)	±28º
Taxa de quadros (profundidade de cor e luz)	30 frames por segundo (FPS)
Resolução, fluxo de profundidade	QVGA (320x240)
Resolução, fluxo de cor	VGA (640x480)
Formato de áudio	16-KHz, 16-bit mono pulse code modulation (PCM)
Características de entrada de áudio	Um conjunto de quatro microfone com 24-bit analógico-digital (ADC) e Processamento de sinal Kinect residente, como cancelamento de eco acústico e supressão de ruído

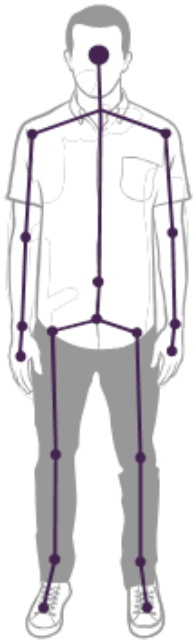
Fonte: Adaptação de Avancini (2012, p. 9)

2.8.2 Detecção de esqueleto

Atualmente o rastreador esquelético do sensor é otimizado para detectar até seis pessoas independente se está em pé ou sentado, desde que esteja de frente para o Kinect. O sensor consegue mapear o esqueleto de duas pessoas simultaneamente durante uma cena, mas possui problemas em relação a partes não visíveis, como em poses laterais em que partes do corpo se encontram cobertas (Kinect For Windows SDK, 2014).

Segundo Cardoso e Schimidt (2012, p.30) “o Kinect possui um processamento interno que utiliza um algoritmo de redes neurais artificiais que mapeia 20 articulações do usuário formando um esqueleto”, como pode ser visualizado pela Figura 9.

Figura 9 - pontos reconhecíveis pelo Kinect.



Fonte: Adaptação de Microsoft (2014, online).

O mapeamento das articulações e formação do esqueleto, permite identificar e manipular os valores dos pontos, nos quais são apresentados sobre a cabeça, pescoço, ombros, cotovelos, pulsos, mãos, tórax, cintura, joelhos, tornozelos e pés.

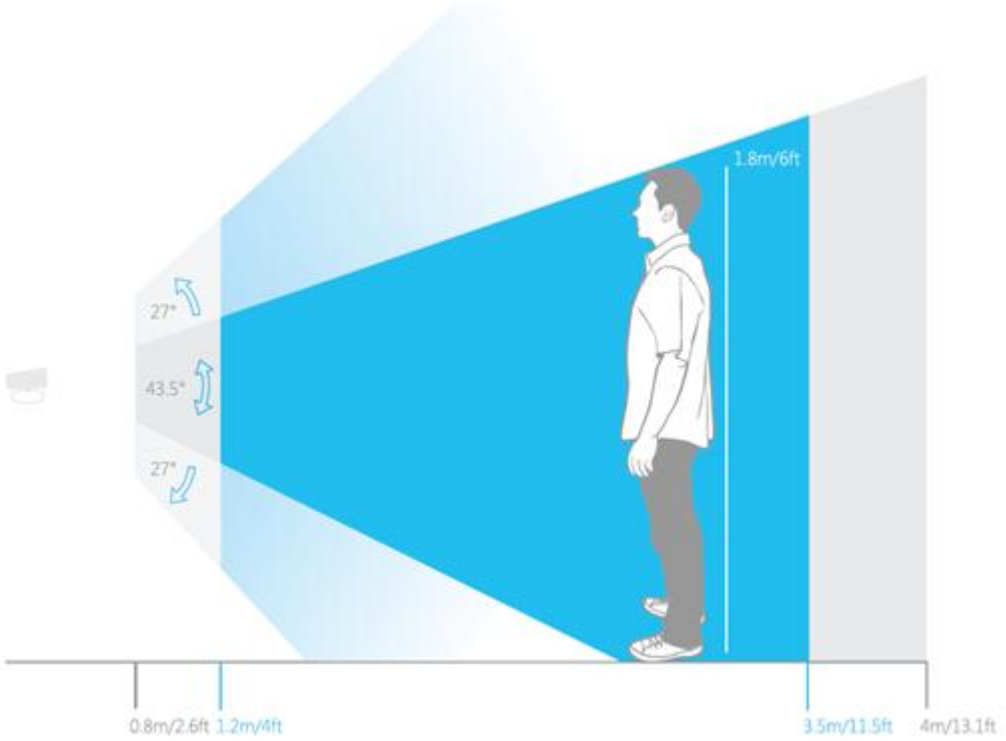
Este esqueleto mapeado possui os pontos formados por coordenadas tridimensionais, ou seja, possui os valores X, Y e Z de cada articulação. Com essas coordenadas se torna possível a geração de imagens em 3D.

2.8.3 Campos de Visão

O sensor possui um campo de visão de 57º na horizontal, 43º na vertical e 70º na diagonal, além da base que é capaz de mover o sensor mais ou menos 27º para cima ou para baixo (AVANCINI, 2012).

O sensor apresenta algumas limitações técnicas em seu funcionamento, onde são mapeados apenas os pontos cuja distância em relação ao Kinect está entre 0,4 m e 4 m dependendo do modo de intervalo. No modo de intervalo padrão o Kinect pode detectar pessoas entre 0,8 m e 4,0 m de distância, mas é sugerido uma distância entre 1,2 m e 3,5 m (*Kinect for Windows SDK: Skeletal Tracking*, 2014), como pode ser visto na Figura 10.

Figura 10 - Campo de visão do Kinect.



Fonte: Adaptação de Microsoft (2014, online).

A Microsoft (2014) recomenda que o sensor seja posto em locais entre 0,6m e 1,8m, mas a posição a ser posta o sensor já não faz muita diferença, desde que a distância esteja dentro do seu campo de visão, pois que pontos fora dessa faixa não são reconhecidos.

Este campo de visão do sensor permite que o dispositivo seja utilizado facilmente em consultórios ou salas de fisioterapia, pois o sensor pode ser posto sobre uma mesa de escritório ou macas.

2.9 Software Development Kit

Um SDK é definido por Campos (2013, p.11) como "um software com um conjunto de ferramentas que possibilitam a criação de aplicações para um determinado produto de software ou hardware". Estes softwares acabam sendo responsáveis por fazer uma interação entre o desenvolvedor e o hardware.

Inicialmente o Kinect foi desenvolvido com a finalidade de ser utilizado exclusivamente como console para o Xbox 360, mas acabou sendo descoberto que era possível utilizar as funções de movimento e voz do dispositivo em um computador por meio da porta USB. A Microsoft vendo este potencial do dispositivo acabou lançando em 2010 o kit de desenvolvimento de *softwares*, o qual permite a utilização de todas as funções do sensor (VARQUEZ, 2013).

Atualmente existem diferentes SDK's que permitem o desenvolvimento de aplicações para o sensor Kinect, sendo em sua grande maioria de livre acesso, nas quais podem ser compatíveis com uma variedade de linguagens e plataformas. Algumas SDKs podem ser conhecidas na tabela 2, na qual faz um comparativo entre os três SDK's mais utilizadas, sendo estas a OpenKinect, a OpenNI e a Microsoft Kinect SDK.

Tabela 2 - Comparação dos Framework.

Características	OpenKinect	OpenNI	Microsoft Kinect SDK
Licença	LGPLv3+	LGPLv3+	Proprietária uso não comercial
Driver	Libfreenect	Sensor Kinect Avin2	Microsoft Research Driver
Plataforma	Windows Linux Mac OS	Windows Linux Mac OS	Windows
Suporte para câmera com sensor PrimeSense	Não	Sim	Não
Kinect's simultâneos	Sim	Sim	Sim
Câmera RGB	Sim	Sim	Sim
Câmera Infravermelho	Sim	Sim	Sim
Projetor Infravermelho	Sim	Sim	Sim
LED	Sim	Sim	Não
Motor Vertical	Sim	Não	Sim
Áudio	Sim	Não	Sim
Detecção de Gestos	Exemplos com OpenCV	NITE Middleware	Nenhum módulo incorporado
Detecção de Esqueleto	Não	Sim	Sim
Detecção de Mão	Não	Sim	Sim
Analisador de Cenas	Não	Sim	Não

Fonte: Adaptação de Apresentação Kinect (ARRAIS, 2012)

Na próxima seção será comentado a respeito de cada uma destas diferentes alternativas de SDK's disponíveis, que permitem a interação com sistemas do tipo RGB-D analisados anteriormente.

2.9.1 OpenKinect

O OpenKinect é uma biblioteca de código aberto que foi desenvolvida por uma comunidade aberta de mais de 2000 membros, com o objetivo de criar a melhor suíte de aplicativos possível para o Kinect (AVANCINI, 2012). Essa SDK permite que sejam desenvolvidas aplicações para o dispositivo Kinect, do qual possui versões para Windows, MacOS X e Linux. E possui *wrappers* que permitem a programação em linguagens como C, C++, Java, C#, Python, Processing entre outras (PAULA, 2011).

Essa SDK utiliza a biblioteca *libfreenect*, a qual fornece as informações de cor, profundidades, motores, acelerômetro e LED existentes no Kinect (YAMADA et al, 2012). Segundo Campos (2013) além de ser um projeto *open source*, desenvolvido em volta do *driver libfreenect*, também contém uma biblioteca que analisa as saídas do sistema, oferecendo uma camada de abstração para segmentos das mãos ou do esqueleto, por exemplo.

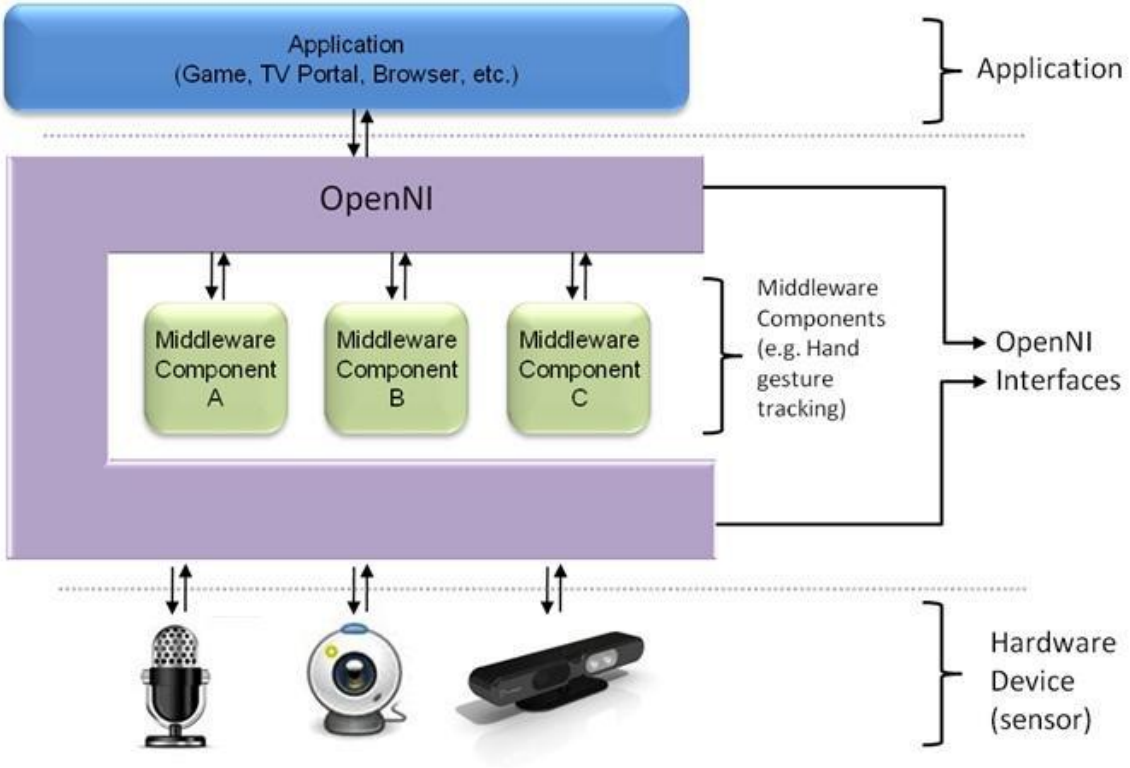
2.9.2 OpenNI

O OpenNI é uma SDK criada pela *Primesense*, empresa israelense fabricante do sensor que serviu de base à *Microsoft* na criação do Kinect. Na qual abriu o código do seu driver e *framework* para dar início à *Open Natural Interaction* (OpenNI), uma organização sem fins lucrativos (PAULA, 2011), onde o principal objetivo da organização é certificar e promover a compatibilidade e interoperabilidade dos dispositivos de Interação Natura (NI), aplicações, middleware e acelerar a introdução de aplicações NI no mercado (AVANCINI, 2012).

Com este objetivo a OpenNI juntamente com sua comunidade *online* desenvolveu um SDK para desenvolvimento de aplicações para dispositivos de interação natural, de modo que fosse compatível com vários sistemas do tipo RGB-D (CAMPOS, 2013). Assim como o OpenKinect, o OpenNI também é uma biblioteca de código aberto e gratuita, sem nenhum tipo de restrição. Sendo um SDK que possui versões para Windows, MacOS e Linux e permite que sejam desenvolvidas aplicações em linguagens como C, C++, Java, C# e Processing (COUTO, 2012; SILVEIRA, 2011).

Conforme Couto (2012) uma API utiliza a *framework* OpenNI, como uma camada intermediária entre a aplicação e os dispositivos de baixo nível, como apresentado na Figura 11.

Figura 11 - Arquitetura de uma API que utiliza o OpenNI.



Fonte: Avancini (2012, p.66).

A figura 11 apresenta uma divisão em três camadas da arquitetura de uma API que utiliza o OpenNI, sendo a camada de cima a que representa os softwares que utilizam as APIs, como aplicações para Games, TV Portal, Browser entre outros.

A camada intermediária fornece uma comunicação entre os componentes de *middleware*, para realizar uma análise e mapeamento das informações geradas pelos dispositivos da camada inferior, na qual recebe as informações do ambiente por meio de sensores, câmeras e microfones.

2.9.3 Kinect for Windows SDK

Esse kit de desenvolvimento de software é a SDK oficial da *Microsoft*, que foi lançada em junho de 2011 com o intuito de permitir o desenvolvimento de aplicações para sistemas Windows 7, possuindo em sua versão Beta uma licença de uso não comercial. Essa SDK acabou permitindo que os desenvolvedores pudessem utilizar o Kinect como uma interface para novos tipos de aplicações (CARDOSO; SCHMIDT, 2012). O Kinect for Windows SDK

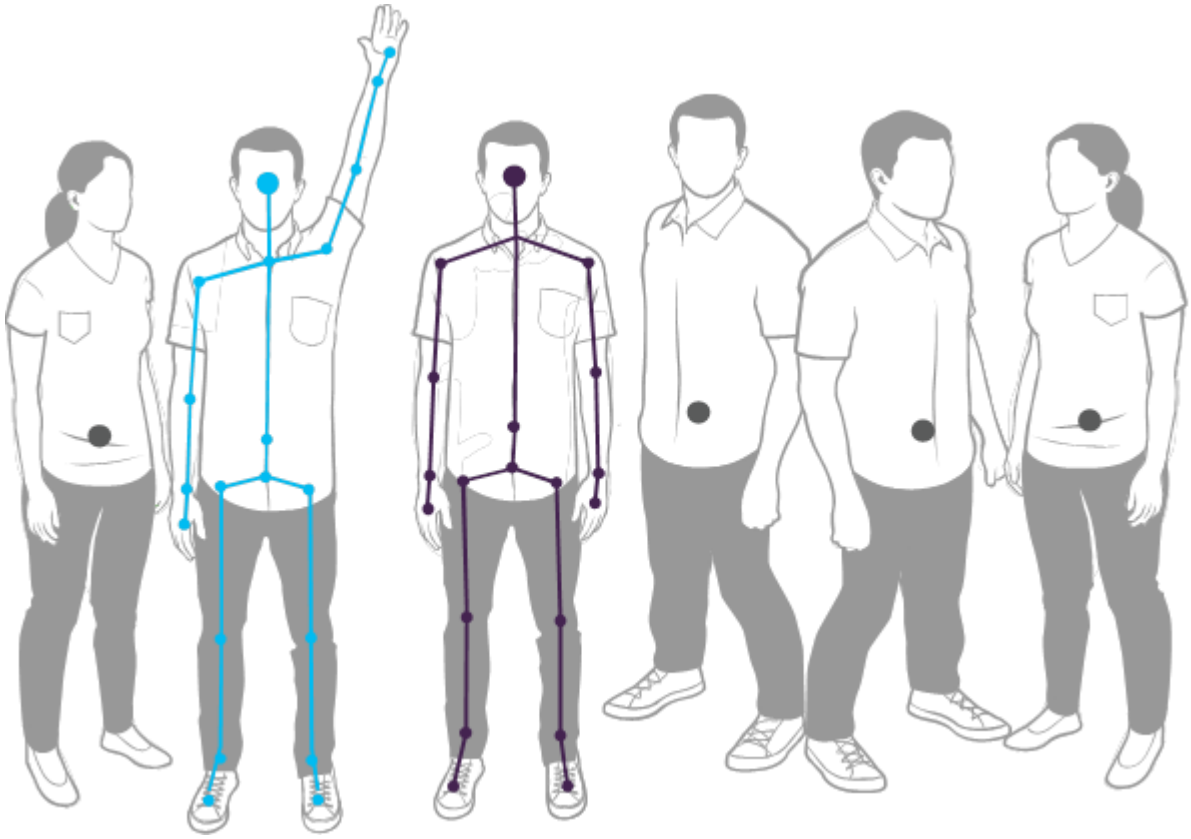
> Suporta o desenvolvimento de aplicações em C++, C# ou Visual Basic, usando o Microsoft Visual Studio 2010 ou 2012. Permite o acesso direto a informação dos diversos sensores existentes na plataforma. O facto de estar a ser desenvolvido pela Microsoft assegura a qualidade da API, assim como uma boa documentação com informação relativa a todas as bibliotecas existentes. Este SDK detecta até 20 articulações, pode seguir até 2 utilizadores em simultâneo e não necessita de uma calibração inicial por parte do utilizador. Recorre a um sistema especializado que compara imagens de corpos humanos com os dados que são recebidos do sensor de profundidade permitindo assim determinar rapidamente formas humanas e, como resultado, detectar a posição das articulações. Contudo é propenso a falsos positivos (CAMPOS, 2013, p.12).

Através da Kinect for Windows SDK podem ser desenvolvidas aplicações que utilizem as informações capturadas pelos sensores (CHERCHIGLIA, 2011), fornecendo acesso às APIs, e permitindo as seguintes funcionalidades (MICROSOFT, 2014):

- Controle direto do sensor;
- Acesso a imagens diretas do sensor de profundidade, e câmera colorida;
- Acesso ao microfone multimatriz;
- Capacidade de processamento de áudio, incluindo supressão de ruídos e cancelamento de eco;
- Capacidade de identificação da fonte emissora do som atual;
- Integração com a API de reconhecimento de fala do Windows;
- Construção de aplicações em C++, C# ou Visual Basic;
- Detecção do esqueleto de uma ou duas pessoas dentro do campo de visão do Kinect para aplicações orientadas a gestos.

Essa SDK permite que sejam detectados até 6 usuários ao mesmo tempo e o rastreamento de 2, possibilitando o uso multiusuário ou apenas a captura de um dos usuários que estejam em frente a câmera (SILVEIRA, 2011). No rastreamento dos esqueletos detectados são capturados 20 pontos, um para cada *joint* do corpo humano (CATUHE, 2012), como pode ser visto na figura 12.

Figura 12 - Reconhecimento e rastreamento de pessoas.



Fonte: Microsoft (2014, online).

Segundo Campos (2013) um ponto a favor deste SDK é o fato de realizar detecções preditiva de articulações, assim mantendo um nível elevado na precisão quando comparado com os outros SDK's. Por possibilitar a utilização total dos recursos contidos no sensor Kinect e com uma maior precisão, esta SDK foi a escolhida para o desenvolvimento da ferramenta a ser entregue.

3 MATERIAIS E MÉTODOS

Nessa seção serão apresentados os materiais utilizados e os procedimentos metodológicos realizados no desenvolvimento da ferramenta.

3.1 Local de Desenvolvimento

Esse trabalho foi desenvolvido no Laboratório de Multimídia (Labmídia), em conjunto com o Laboratório de Tecnologia em Saúde (LTS), ambos localizados no Complexo de Informática do Centro Universitário Luterano de Palmas (CEULP/ULBRA). Como requisito parcial para a disciplina de Trabalho de Conclusão de Curso em Ciência da Computação II (TCC II), realizada durante o primeiro semestre de 2015, e estando vinculado ao grupo de Pesquisa Tecnologia, Saúde e Qualidade de Vida, no qual realiza pesquisas com a inclusão do Sensor Microsoft Kinect no auxílio de tarefas fisioterapêuticas.

Para os testes e comparação da ferramenta foi utilizada uma amostra de 15 voluntários, colegas de laboratório, sendo estes do sexo masculino, contendo entre 17 e 40 anos, todos alunos do curso de Sistema de Informação do Centro Universitário Luterano de Palmas (CEULP/ULBRA), nos quais se ofereceram para participar.

3.2 Materiais

Nesta seção apresenta-se os materiais utilizados para realização do trabalho.

3.2.1 Fontes Bibliográficas

A princípio, foram realizados estudos sobre os conceitos envolvidos no trabalho para se chegar aos artigos de periódicos científicos, dissertações de mestrado e teses de doutorado, livros entre outros materiais que exponham conhecimento sobre o tema. As fontes bibliográficas utilizadas são apresentadas em Referências (seção 5).

3.2.2 Hardwares

Durante o processo de desenvolvimento da ferramenta foi utilizado um computador com os requisitos mínimos de *hardware*, contendo um processador dual core 2,66 GHz de 32 bits, barramento USB 2.0 dedicado, 2 GB de RAM e placa de vídeo compatível com o DirectX 9.0c, assim como um sensor *Microsoft Kinect for Windows* do XBOX 360 com adaptador do sensor Kinect para computador.

3.2.3 Softwares

Os *softwares* utilizados no desenvolvimento do trabalho foram:

- IDE Visual Studio 2013 para realização de alterações na codificação da aplicação, na qual foi implementada na linguagem de programação orientada a objetos C# (C Sharp). Para construção da interface de apresentação foi utilizado o padrão *Windows Presentation Foundation* (WFP), um tipo de projeto disponibilizado pelo Visual

Studio na construção de aplicações Desktop para Windows que utiliza o recurso *Extensible Application Markup Language* (XAML).

- SQL SERVER 2012, utilizado para armazenamento dos dados.
- LINQ to SQL, fará a conexão entre a aplicação e o banco de dados, uma infraestrutura fornecida pelo Visual Studio para gerenciamento do banco de dados relacionais em tempo de execução.
- *Kinect for Windows Software Development Kit* (SDK) na versão 1.8, sendo necessário a instalação e adiciona-lo às referências do projeto para comunicação entre o sensor kinect e a aplicação.
- *Kinect for Windows Developer Toolkit* na versão 1.8, para disponibilizar recursos e simplificar no desenvolvimento da aplicação que utiliza o sensor Kinect, sendo necessário assim como o SDK a sua instalação e adição às referências do projeto.

3.3 Metodologia

Para o desenvolvimento deste trabalho, utilizou-se como desenho de estudo uma pesquisa com finalidade metodológica aplicada, objetivo metodológico exploratório e natureza quantitativa, sendo realizado em laboratório com procedimento quase-experimental em um estudo piloto para teste de acurácia.

O processo de desenvolvimento deste trabalho se dividiu em 5 etapas de fundamental importância para que o mesmo fosse concluído com êxito, sendo estas etapas representadas pela Figura 13 apresentada a seguir.

Figura 13 - Metodologia do projeto.

Como apresentado na Figura 13, inicialmente foi necessária a realização de estudos sobre os temas relacionados (1), fundamentando-se em conhecimentos bibliográficos encontrados em artigos de periódicos científicos, dissertações de mestrado, teses de doutorado, livros entre outros materiais que exponham conhecimento sobre o tema. Esta etapa teve fundamental importância para conhecer o sensor Kinect e entender o processo de avaliação da Amplitude de Movimento.

A partir dos estudos realizados na primeira etapa, foi possível identificar as tecnologias e ferramentas a serem utilizadas no desenvolvimento da ferramenta, bem como definir os objetivos a serem alcançados, a hipótese e a justificativa para a realização do trabalho. Estas atividades foram documentadas na etapa de Elaboração do Projeto (2), onde também foram consolidados os métodos a serem utilizados, permitindo prosseguir com o desenvolvimento de uma ferramenta para captura de ângulos corporais de ADM.

Para realização da etapa de desenvolvimento da ferramenta (3), foram utilizadas as tecnologias definidas pela etapa anterior, que contou com a utilização de um sensor *Microsoft Kinect for Windows*, no qual proporcionou ao sistema uma visão computacional sobre o ambiente através dos seus *hardwares*, utilizando sua câmera RGB e seu emissor e receptor de raios infravermelhos durante a coleta dos dados.

Para identificação do paciente e utilização dos recursos do sensor Kinect foi instalado o driver *Kinect for Windows Software Development Kit*, que permitiu o mapeamento das coordenadas corporais do paciente e geração do esqueleto de pontos articulares por meio das classes `ColorStream`, `DepthStream` e `SkeletonStream`.

A integração desses recursos e desenvolvimento da ferramenta se realizou dentro da IDE Visual Studio, onde foi utilizado a linguagem C# para implementação, o padrão WPF na construção da interface e o banco de dados SQL conectado na aplicação por meio do LINQ to SQL.

Após a etapa de desenvolvimento da ferramenta, foram realizados testes informais, com o intuito de identificar e corrigir erros no processo de avaliação e nos métodos utilizados no processo de obtenção dos dados, para que pudesse dar início a etapa seguinte.

A quarta etapa consistiu no processo de testes e comparação da ferramenta com goniometria (4), para identificar se a ferramenta disponibiliza dados uteis à serem utilizados no processo de avaliação da Amplitude de Movimento. Para isto, foi realizado a comparação dos resultados obtidos pela aplicação como os resultados obtidos através da goniometria, como apresentado na seção 4.3.

A quinta etapa consistiu no processo de escrita dos resultados (5), onde foram documentados todos os procedimentos realizados durante desenvolvimento da ferramenta. Descrevendo de forma detalhada a implementação, os testes e comparação, apresentados na seção seguinte através de imagens, trechos de códigos e tabelas.

Ao término deste trabalho, esta aplicação foi vinculada às demais aplicações já desenvolvidas pelo grupo de Pesquisa Tecnologia, Saúde e Qualidade de Vida, se tornando

um módulo dentro de um projeto maior que busca utilizar o sensor Kinect no auxílio de tarefas fisioterapêuticas.

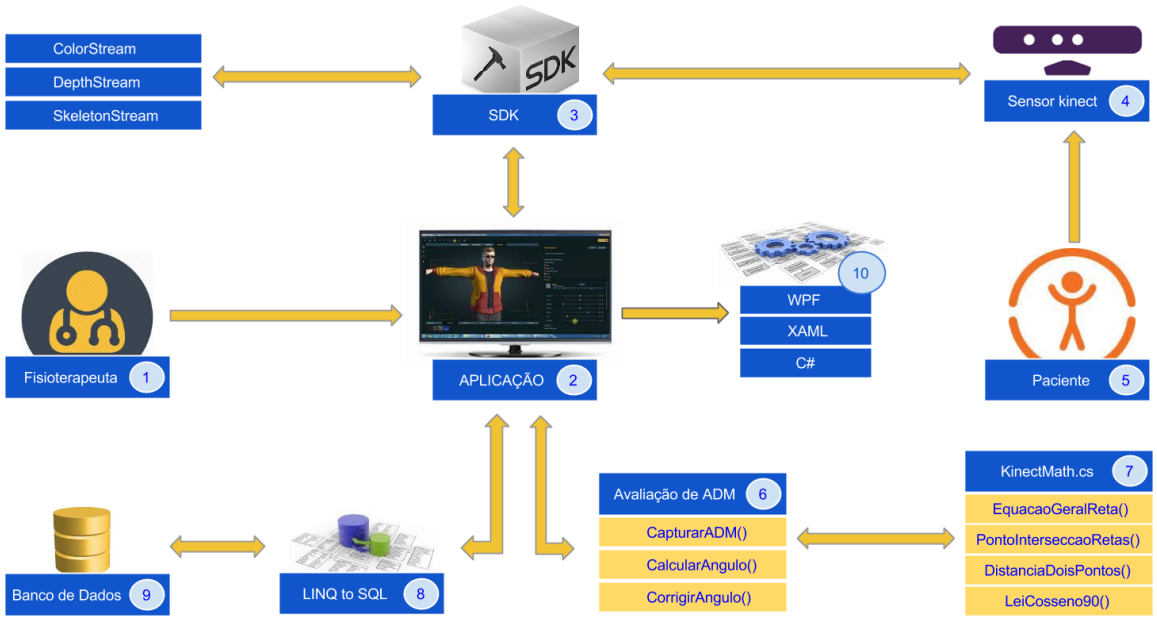
4 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados obtidos na realização do trabalho, que se propôs a desenvolver uma ferramenta que capturasse os ângulos corporais de ADM em articulações dos ombros e quadris, realizados em plano coronal de visão anterior utilizando o sensor Kinect. Inicialmente será apresentada uma visão geral da Ferramenta, apresentando sua arquitetura e o funcionamento da mesma (seção 4.1). Em seguida, apresenta-se o desenvolvimento da aplicação (seção 4.2), e os resultados obtidos nos testes e comparação da ferramenta (seção 4.3).

4.1 Visão Geral da Ferramenta

Para entender melhor a estrutura e o funcionamento da ferramenta, foi elaborada a arquitetura da mesma, como apresentada a seguir (Figura 14).

Figura 14 - Arquitetura da Aplicação.



Como apresentado pela figura 14, o funcionamento da ferramenta segue os seguintes passos:

- O fisioterapeuta (1) interage com a aplicação (2), selecionando a articulação que será avaliada e iniciando o processo de captura da ADM do movimento realizado pelo paciente (5). Sendo a aplicação capaz de monitora em tempo real o ângulo do movimento realizado pelo paciente que deverá estar posicionado a frente do sensor Kinect.

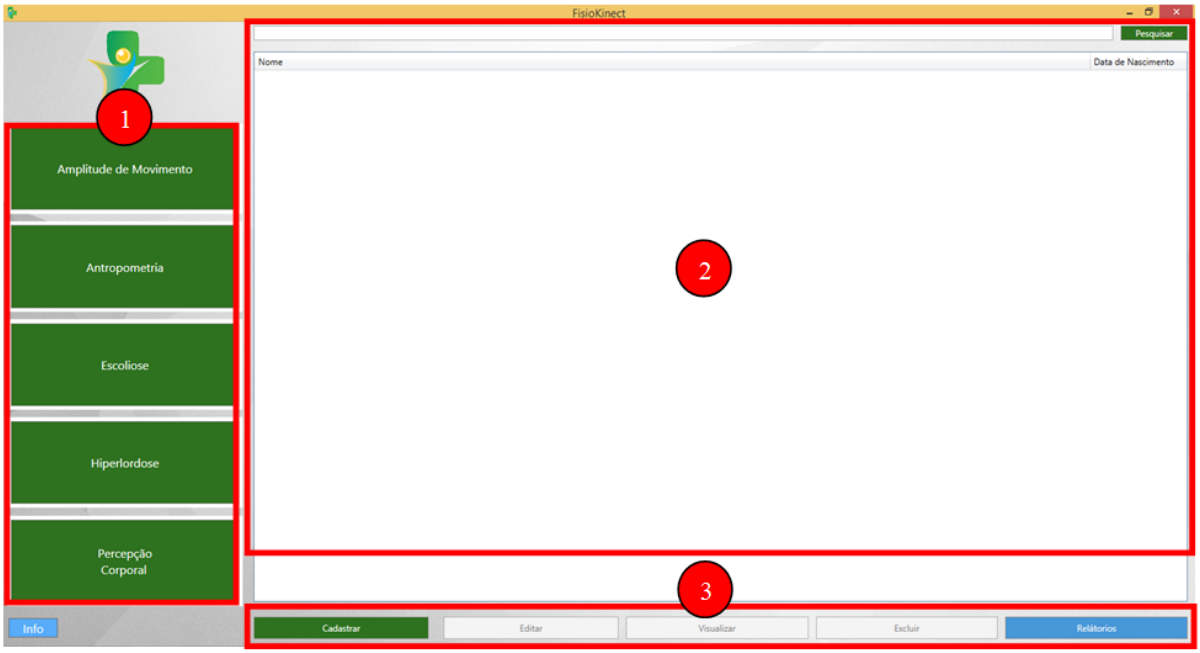
- O paciente executará cada movimento três vezes, capturando a ADM em cada um deles, e ao fim o resultado é apresentado à tela do fisioterapeuta, informando à ADM dos três movimentos realizados e uma média das três mensurações.
- Esta ADM é obtida através da utilização da câmera RGB, do emissor e receptor de raios infravermelhos contidos no sensor Kinect (4), que por sua vez, é conectado à aplicação por meio dos drivers existentes no SDK (3).
- O SDK permite além da conexão entre a aplicação e o sensor kinect, o mapeamento e criação de um esqueleto do paciente, fornecendo valores das coordenadas x, y e z das articulações, denominadas *Joints.* Estes valores são processados pelos métodos de Avaliação de ADM (6), que utiliza as fórmulas matemáticas contidas na classe KinectMath (7) para retornar o ângulo da ADM.
- Os dados gerados na avaliação são inseridos em um banco de dados SQL (9), que é conectado à aplicação por meio do gerenciador de banco de dados relacionais LINQ to SQL (8). A aplicação conta também com as ferramentas (10): WPF e XAML na construção da interface gráfica para aplicação e da linguagem C# na implementação.

4.2 Implementação da Ferramenta

Esta seção apresenta o processo de implementação da ferramenta assim como os resultados obtidos. Para isto, serão apresentadas imagens e trechos de códigos que demonstram a estrutura, interfaces e métodos utilizados.

Inicialmente, foi desenvolvida a interface de inicialização da ferramenta, sendo definido um padrão único na organização de todas as aplicações desenvolvidas pelo grupo de Pesquisa Tecnologia, Saúde e Qualidade de Vida. Com a organização das aplicações a interface inicial apresenta a seguinte aparência (Figura 15).

Figura 15 - Interface Inicial da Ferramenta.



Como apresentado na Figura 15, esta interface apresenta em (1), os módulos desenvolvidos pelo grupo de pesquisa, os quais foram acoplados a uma única aplicação sendo separados em botões com as funcionalidades pertinentes aos seguintes trabalhos:

- Amplitude de Movimento - Avaliação da amplitude de movimento (ADM) em plano coronal anterior utilizando o sensor Kinect: articulações do ombro e do quadril.
- Antropometria - Utilizando o Microsoft Kinect na obtenção de atributos antropométricos.
- Hiperlordose - Análise postural computadorizada para identificação de hiperlordose utilizando o kinect.
- Escoliose - Avaliação postural computadorizada utilizando o kinect.
- Percepção Corporal - Automatização do *Image Marking Procedure* para análise do esquema corporal utilizando o Kinect.

A interface apresenta em (2) um campo de busca para pesquisas de pacientes em uma lista contendo todos os pacientes cadastrados no sistema. Logo abaixo (3), são listados os botões de cadastro, edição, visualização e exclusão de pacientes, além do botão para geração de relatórios.

Ao selecionar o paciente a ser avaliado e acessar o módulo `Amplitude de Movimento`, para avaliação da amplitude de movimento localizado em (1), a aplicação apresentará os componentes visuais da interface de avaliação e inicializará o sensor Kinect. É válido ressaltar que cada aplicação que utiliza o sensor Kinect acaba possuindo elementos

básicos em seu funcionamento. Por exemplo, inicialmente, deve-se detectar o sensor e em seguida realizar a inicialização do mesmo, após inicializado o sensor produzirá dados que são habilitados e processados pela aplicação (WEBB; ASHLEY, 2012).

Conforme apresentado, aplicações que utilizam o sensor Kinect devem iniciar um objeto `KinectSensor`, para representar o sensor Kinect e permitir acesso a uma fonte *multstream* para fornecimento de fluxos de dados, como apresentado no método `InicializarSensor()` (Figura 15).

Figura 16- Inicialização do Sensor kinect.

```
47      public void InicializarSensor(KinectSensor kinectS)
48      {
49          kinect = kinectS;
50      }
```

Esta fonte permite a criação de um fluxo para cada tipo de dado que recolhe, criando métodos de fluxos de cores de imagem (`ColorImageStream`), profundidade (`DepthImageStream`) e esqueleto (`SkeletonStream`), apresentados a seguir.

4.2.1 Fluxo de Cores de Imagem

O sensor Kinect possui em sua arquitetura uma câmera de cores com uma série de configurações, permitindo o processamento de diferentes formatos de imagem e influenciando diretamente na qualidade da imagem e na quantidade de quadros processados por segundo. O fluxo de imagem colorida é denominado `ColorStream`, onde este fluxo possui as seguintes opções de formato:

- `RgbResolution640x480Fps30`
- `RgbResolution1280x960Fps12`
- `YuvResolution640x480Fps15`
- `RawYuvResolution640x480Fps15`
- `RawBayerResolution640x480Fps30`
- `RawBayerResolution1280x960Fps12`

Por padrão, o formato utilizado pela aplicação é o `RgbResolution640x480Fps30`, no formato RGB (*Red, Green e Blue*) cada *pixel* utiliza uma quantidade de cor vermelha, uma quantidade de cor verde e uma quantidade de cor azul, onde a soma dos três valores representam a cor do *pixel*. O formato 640x480 representa que a imagem possui 640 colunas de *pixels* por 480 linhas de *pixels* organizados em formato de matriz, com a intensidade de 30 *Fps (Frames per second),* sendo o sensor capaz de disponibilizar à aplicação 307200 *pixels* por *Frame* (CATUHE, 2012).

Segundo Webb e Ashley (2012), para se trabalhar com o fluxo de cores de imagens deve se executar três etapas, onde primeiramente deve-se habilitar o fluxo de cores de imagem, como apresentado no método `InicializarSensor()` (Figura 16).

Figura 17 - Habilitação do fluxo de imagem.

```
47      public void InicializarSensor(KinectSensor kinectS)
48      {
49          kinect = kinectS;
50          kinect.ColorStream.Enable();
51      }
```

Conforme ilustrado pela Figura 16, a habilitação do fluxo de cores de imagem acontece por meio do objeto `Kinect`, no qual permite acesso ao fluxo de cores de imagem através da classe `ColorStream`, que instanciada como o método `Enable()` sem passagem de formatos por parâmetro utiliza o seu formato padrão na obtenção das imagens.

A segunda etapa consiste em como a aplicação deve identificar e extrair os dados dos fluxos de imagem, sendo assim criado o método `CameraImagemRGB()` ilustrado pela Figura 17. Este método é responsável por extrair os dados do fluxo de cores de imagem disponibilizados pela câmera de imagens RGB e transforma-los em *bytes* de imagem como apresentado a seguir.

Figura 18 - Método CameraImagemRGB.

```
134     private byte[] CameraImagemRGB(ColorImageFrame quadro)
135     {
136         if (quadro == null) return null;
137         using (quadro)
138         {
139             byte[] bytesImagem = new byte[quadro.PixelDataLength];
140             quadro.CopyPixelDataTo(bytesImagem);
141             return bytesImagem;
142         }
143     }
```

O método apresentado pela Figura 17, se inicia com a verificação do objeto `ColorImageFrame` denominado `quadro` (linha 136), controlando se o objeto se encontra vazio, caso seja verdade o método retornara *null*, realizando um controle de exceção do método. Na linha 139, é declarado um *array* de *byte* `bytesImagem` que recebe um novo objeto contendo a quantidade de *pixels* existentes no *frame* recebido pela propriedade `PixelDataLength`.

Logo em seguida, linha 140, são copiados os dados do *frame* para o `bytesImagem`, para no final, linha 141, o `bytesImagem` retorne os *bytes* da imagem para o método `Kinect_TodosQuadros()`.

A terceira etapa consiste no processamento dos dados da imagem, sendo constantemente extraído e processados os dados enquanto possuir dados de imagem disponível. O método `Kinect_TodosQuadros()` é executado a cada novo *Frame* que estiver disponível como apresentado pela Figura 18.

Figura 19 - Método Kinect_TodosQuadros.

```
187         // Obter todos os quadros de um conjunto de Frames obtidos da Camera RGB
            1 reference
188         private void kinect_TodosQuadros(object sender, AllFramesReadyEventArgs e)
189         {
190             byte[] imagem = CameraImagemRGB(e.OpenColorImageFrame());
191             if (imagem != null)
192             {
193                 imageKinect.Background = new ImageBrush(BitmapSource.Create(kinect.ColorStream.FrameWidth,
194                     kinect.ColorStream.FrameHeight, 96, 96, PixelFormats.Bgr32, null, imagem,
195                     kinect.ColorStream.FrameWidth * kinect.ColorStream.FrameBytesPerPixel));
196             }
197             imageKinect.Children.Clear();
198             DesenharEsqueletoUsuario(e.OpenSkeletonFrame());
199             CapturarADM();
200         }
```

Como apresentado, o método `Kinect_TodosQuadros()` é responsável por obter todos os quadros de um conjunto de *Frames* disponibilizados pela câmera, sendo realizada a chamada do método `CameraImagemRGB()` (linha 190), que instancia os *bytes* que compõem o *frame* ao *array* de *byte* denominado `imagem`. Na linha 191, é verificado se o *array* de *bytes* `imagem` não se encontra vazio ou nulo, controlando para que se a afirmação for verdadeira o atributo *background* do objeto `canvas` seja preenchido pela imagem no formato *bitmap* (linha 193).

Na linha 197, se instancia o método `Children` do objeto `canvasKinect` como `Clear()`, para que limpe os componentes a cada vez que o método for chamado, apagando os rastros do usuário e redesenhando a cada quadro processado.

Na linha 198, é chamado o método `DesenharEsqueletoUsuario` passando como parâmetro o quadro de dados do esqueleto, para que seja desenhado o esqueleto do paciente capturado.

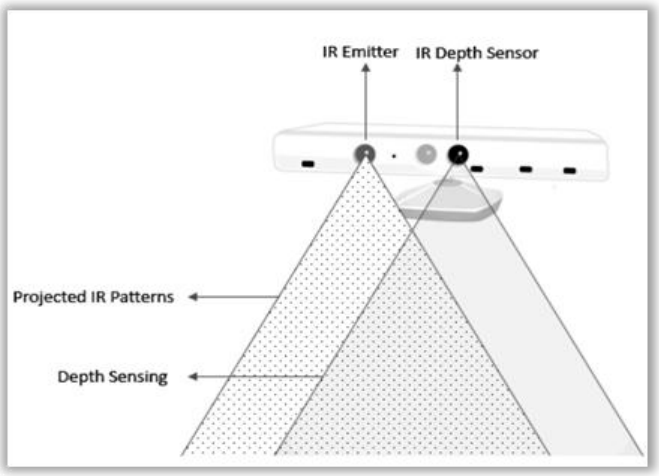
Ao final, linha 199, se chama o método `CapturarADM()`, método responsável por realizar a avaliação da ADM do paciente posicionado a frente do sensor.

Após a habilitação do fluxo de cores de imagem e processamento dos dados obtidos através dos métodos `CameraImagemRGB()` e `Kinect_TodosQuadros()`, foi habilitado o fluxo de profundidade do sensor como apresentado na seção seguinte (seção 4.2.2).

4.2.2 Fluxo de Profundidade

Da mesma forma que o sensor Kinect possui uma câmera RGB para a transmissão do fluxo de cores de imagem, possui em sua arquitetura um emissor de raios infravermelho *(IR Emitter)* e uma câmera de profundidade infravermelho *(IR Depth Sensor)*, utilizados na captura dos dados de profundidade dos objetos, conforme apresentado a seguir pela figura 3 (MICROSOFT, 2015).

Figura 20 - Campo de visão dos dados de profundidade do Kinect.



Fonte: JANA, 2012, p. 11

Segundo Cardoso (2013), o fluxo de profundidade do Kinect também possui diferentes tipos de formatos que funcionam assim como ao sensor RGB, onde se informa por parâmetro a resolução utilizada no método *Enable* da classe do fluxo de profundidade `DepthStream`, que pode ser de 640 × 480, 320 x 240 ou 80 x 60, sendo a primeira a resolução padrão. Porém no fluxo de profundidade só se pode alterar a resolução, mantendo-se os formatos padrão de 30 *Fps* a 16 *bits* por *pixel*.

O sensor de profundidade do Kinect também pode trabalhar em dois modos de captura para os valores de profundidade, podendo escolher uma das opções de distância do espaço de leitura do ambiente. No "modo padrão" (*Default Mode*), a distância de captura varia entre 0,8m e 4m da posição do objeto a ser capturado em relação ao sensor. Já no "modo de perto" (*Near Mode*), a distância deve ser entre 0,5m e 3m, além de ser um modo que afeta diretamente a precisão do fluxo de profundidade (CATUCHE, 2012).

Para Webb e Ashley (2012), trabalhar com a classe de fluxo de dados de profundidade `DepthImageStream` se assemelha a classe `ColorImageStream`, pois as duas classes possuem a mesma classe pai (`ImageStream`). Dentro das classes responsáveis pelo

processamento dos dados de profundidade também se destaca a classe `DepthImageFrame`, na qual contém um buffer que guarda os dados de profundidade de cada frame, como dimensões, formato e dados de pixel, além de permitir o mapeamento de coordenadas entre os quadros de esqueleto e de cores.

O fluxo de profundidade e representado pela classe `DepthStream`, que é responsável por realizar funções básicas do sensor IR e obter os dados de profundidade, sendo estes dados muito importante no processo de reconhecimento do ambiente. Para utilizar o fluxo de profundidade foi acrescentada mais uma linha ao método InicializarSensor(), como pode ser visto pela Figura 20.

Figura 21 - Habilitação do fluxo de profundidade.

```
47      public void InicializarSensor(KinectSensor kinectS)
48      {
49          kinect = kinectS;
50          kinect.ColorStream.Enable();
51          kinect.DepthStream.Enable();
52      }
```

Assim como o fluxo de cores de imagem a habilitação do fluxo de profundidade é bem simples, sendo novamente utilizado o objeto `Kinect` que utiliza a classe `DepthStream` para acesso ao fluxo de dados da imagem, taxa de quadros e resolução, através da instanciação do método como `Enable()`. Como apresentado, linha 51, o método foi instanciado sem definição de formato passado por parâmetro, sendo assim utilizado o formato padrão apresentado anteriormente.

É importante ressaltar que os dados de profundidade são muito importantes em aplicações que utilizam o sensor kinect para identificação de ambientes, objetos e pessoas, pois através do fluxo de profundidade se torna possível identificar ambientes, objetos e mapear pessoas, através do fluxo de esqueleto. O fluxo de esqueleto será apresentado a seguir, demostrando suas características e utilização no trabalho.

4.2.3 Fluxo de Esqueleto

O fluxo de esqueleto se torna semelhante aos demais já apresentados, porém, não conta com um sensor físico responsável por sua criação. Este fluxo representado pela classe `SkeletonStream` é gerado por meio de um conjunto de processamentos e de sensores que permitem a identificação de usuários.

Para habilitação do fluxo de esqueleto, foi realizado um procedimento semelhante aos demais fluxos, como pode ser apresentado pela Figura 21.

Figura 22 - Habilitação do fluxo de esqueleto.

```
47      public void InicializarSensor(KinectSensor kinectS)
48      {
49          kinect = kinectS;
50          kinect.ColorStream.Enable();
51          kinect.DepthStream.Enable();
52          kinect.SkeletonStream.Enable();
53      }
```

Como ilustrado pela Figura 21, foi inserida a linha 52, na qual realiza a habilitação do fluxo de esqueleto através da instanciação da classe `SkeletonStream` como Enable(), porém, para uma maior qualidade no processo de captura e minimizar a tremulação dos pontos obtidos pelo sensor, foi acrescentado ao método `InicializarSensor()` um filtro de suavização *"smoothing filter"* como apresentado pela Figura 22.

Figura 23 - Método InicializarSensor.

```
47      public void InicializarSensor(KinectSensor kinectS)
48      {
49          kinect = kinectS;
50
51          TransformSmoothParameters smoothingParam = new TransformSmoothParameters();
52          {
53              smoothingParam.Smoothing = 0.75f;
54              smoothingParam.Correction = 0.1f;
55              smoothingParam.Prediction = 0.1f;
56              smoothingParam.JitterRadius = 0.05f;
57              smoothingParam.MaxDeviationRadius = 0.1f;
58          };
59
60          kinect.ColorStream.Enable();
61          kinect.DepthStream.Enable();
62          kinect.SkeletonStream.Enable(smoothingParam);
63          kinect.AllFramesReady += kinect_TodosQuadros;
64      }
```

Como ilustrado, para realizar ajustes nos quadros e minimizar a tremulação dos pontos e valores das coordenadas, utilizou-se um filtro de suavização criado através do objeto `smoothingParam` da classe `TransformSmoothParameters` (linha 51). Em seguida, foram atribuídos em sua configuração os valores de 0.75f para o atributo `Smoothing` (linha 53), 0.1f para `Correction` (linha 54), 0.1f para `Prediction` (linha 55), 0.05f para `JitterRadius` (linha 56) e 0.1f para `MaxDeviationRadius` (linha 57).

A habilitação do fluxo de profundidade `SkeletonStream`, linha 62, foi alterada, onde foi passando o objeto `smoothingParam` como parâmetro do método `Enable` para atribuir a configuração de *smoothing* ao fluxo de esqueleto.

Após a inicialização dos três fluxos da classe `KinectSensor` foi utilizada a classe `AllFramesReady` (linha 64) para processar as informações dos três tipos de fluxos de dados, retornados do método `Kinect_TodosQuadros`, que como apresentado pela Figura 18 é responsável por obter todos os quadros de um conjunto de *Frames*.

O método `Kinect_TodosQuadros` também realiza a chamada do método `DesenharEsqueletoUsuario()`, utilizado para desenhar os ossos e articulações do esqueleto do paciente. Este método pode ser visualizado pela Figura 24 apresentada a seguir.

Figura 24 - Método DesenharEsqueletoUsuario.

```
202         //Desenhar o Esqueleto do Usuário
            1 reference
203         private void DesenharEsqueletoUsuario(SkeletonFrame quadro)
204         {
205             if (quadro == null) return;
206             using (quadro)
207                 quadro.DesenharEsqueletoUsuario(kinect, imageKinect);
208         }
```

Como apresentado, o método `DesenharEsqueletoUsuario()` recebe um objeto `quadro` da classe `SkeletonFrame` contendo o esqueleto obtido pelo sensor Kinect (linha 203). Em seguida, linha 205, é usado um condicional como controle de exceção, sendo verificado se o quadro de esqueleto passado por parâmetro se encontra vazio ou *null*.

Na linha 206, é utilizado o objeto `quadro` na chamada do método `DesenharEsqueletoUsuario()` da classe `Extensao`. Esta classe juntamente com a classe `EsqueletoUsuario` possue um conjunto de métodos que são responsáveis por manipular as informações visuais do esqueleto. Na utilização destas classes são passadas as informações do fluxo de esqueleto através do objeto `kinect`, e um objeto do tipo `canvas` onde será apresentado as informações visuais resultante da chamada do método `DesenharEsqueletoUsuario()`.

Segundo Shotton et al. (2013), o fluxo de esqueleto utiliza a imagem de profundidade representada pela classe `DepthStream`, com a utilização de inteligência artificial gerada através de uma rede neura artificial treinada para interpretar e identificar as partes do corpo humano. Com isso este fluxo consegue identificar as coordenadas X, Y e Z de usuários e suas articulações, mesmo que possuam diferentes tipos e tamanhos de esqueleto.

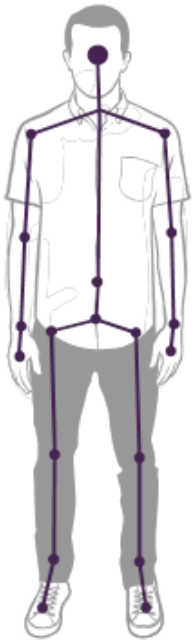
Conforme Cardoso (2013), a Microsoft utilizou o conceito de *Motion Capture* (mocap) de diferentes tipos durante o treinamento desta rede neura artificial, permitindo que este fluxo fosse utilizado para criação de aplicações que se baseiam em movimentos, sendo considerado

muito importante quando se trata de que a maioria das aplicações que utilizam o kinect e feito baseado em movimentos.

Atualmente o rastreador esquelético do sensor é otimizado para detectar até seis pessoas independente se está em pé ou sentado, desde que esteja de frente para o Kinect. O sensor consegue mapear o esqueleto de duas pessoas simultaneamente durante uma cena, mas possui problemas em relação a partes não visíveis, como em poses laterais em que partes do corpo se encontram cobertas (Kinect For Windows SDK, 2014).

Segundo Cardoso e Schimidt (2012) o Kinect possui um processamento interno representado pela classe `SkeletonStream` que permite o mapeamento de 20 pontos do usuário formando um esqueleto que representa as articulações do corpo humano, como pode ser visualizado pela Figura 4.

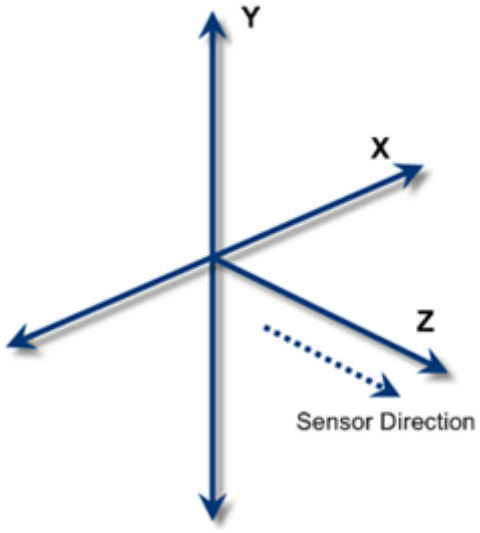
Figura 25 - Pontos reconhecíveis pelo Kinect.



Fonte: Adaptação de Microsoft (2014, online).

O esqueleto do usuário rastreado e representado pela classe `Skeleton` que retorna as articulações do esqueleto por meio da propriedade *Joint,* assim como demonstrado pela Figura 4. O fluxo de esqueleto possui também a classe `SkeletonPoint` na qual representa as coordenadas X, Y e Z de um ponto do esqueleto ou *Joint*, como pode ser apresentada pela Figura 5, que demonstra o sistema de coordenadas do esqueleto em plano tridimensional.

Figura 26 - Sistema de coordenadas 3D.



Fonte: Microsoft (2015, online).

Segundo Catuhe (2012), as coordenadas fornecidas pelo Kinect possuem o eixo x que se estende para a direita do ponto de vista do utilizador, e o eixo y se estende para cima, sendo estas coordenadas expressas em metros, já o eixo z é orientada a partir do sensor para o utilizador, sendo as coordenadas fornecidas em milímetros.

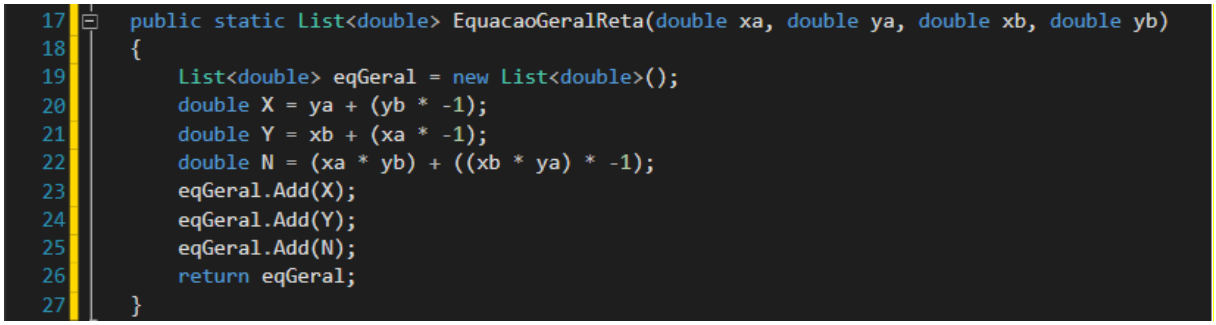
Após a inicialização do sensor kinect e habilitação dos fluxos de cores de imagem, profundidade e esqueleto, serão apresentados os métodos utilizados na aplicação das equações matemáticas (seção 4.2.4) e o processo utilizado na avaliação da amplitude de movimento (seção 4.2.5).

4.2.4 Equações Matemáticas

Para organização do código e estrutura da aplicação os métodos responsáveis pela realização dos cálculos das equações foram agrupados na classe `KinectMath`, sendo esta responsável por conter todos os algoritmos de equações matemáticas utilizadas pela aplicação.

Seguindo o processo de mensuração dos ângulos articulares apresentados pela seção 2.6 foram criados quatro métodos. O primeiro método foi o responsável por realizar o cálculo da equação geral da reta, obtida através de dois pontos. Este método é apresentado na Figura 27.

Figura 27 - Método EquacaoGeralReta.

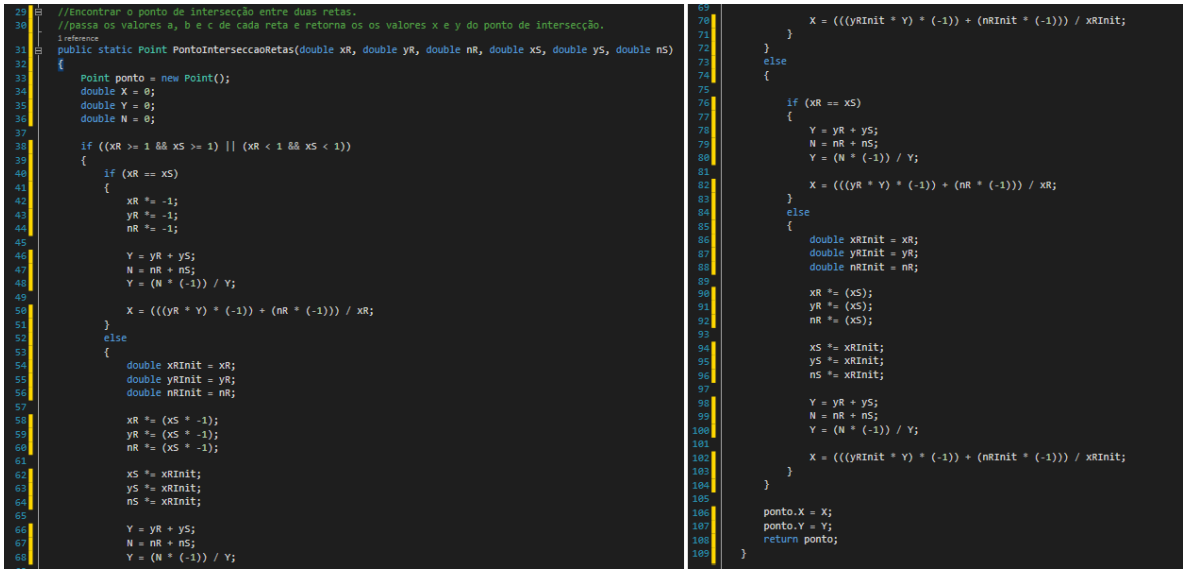


```
17  public static List<double> EquacaoGeralReta(double xa, double ya, double xb, double yb)
18  {
19      List<double> eqGeral = new List<double>();
20      double X = ya + (yb * -1);
21      double Y = xb + (xa * -1);
22      double N = (xa * yb) + ((xb * ya) * -1);
23      eqGeral.Add(X);
24      eqGeral.Add(Y);
25      eqGeral.Add(N);
26      return eqGeral;
27  }
```

Como apresentado pela Figura 27, o método `EquacaoGeralReta()`, recebe os valores das coordenadas x e y de dois pontos, realizando o cálculo e retornando uma lista contendo três valores correspondentes aos valores de a, b e c da equação (1).

O segundo método desenvolvido foi o responsável por obter o ponto de intersecção entre duas retas apresentado pela Figura 28.

Figura 28 - Método PontoInterseccaoRetas.



```
//Encontrar o ponto de intersecção entre duas retas.
//passa os valores a, b e c de cada reta e retorna os os valores x e y do ponto de intersecção.
1 reference
public static Point PontoInterseccaoRetas(double xR, double yR, double nR, double xS, double yS, double nS)
{
    Point ponto = new Point();
    double X = 0;
    double Y = 0;
    double N = 0;

    if ((xR >= 1 && xS >= 1) || (xR < 1 && xS < 1))
    {
        if (xR == xS)
        {
            xR *= -1;
            yR *= -1;
            nR *= -1;

            Y = yR + yS;
            N = nR + nS;
            Y = (N * (-1)) / Y;

            X = (((yR * Y) * (-1)) + (nR * (-1))) / xR;
        }
        else
        {
            double xRInit = xR;
            double yRInit = yR;
            double nRInit = nR;

            xR *= (xS * -1);
            yR *= (xS * -1);
            nR *= (xS * -1);

            xS *= xRInit;
            yS *= xRInit;
            nS *= xRInit;

            Y = yR + yS;
            N = nR + nS;
            Y = (N * (-1)) / Y;

            X = (((yRInit * Y) * (-1)) + (nRInit * (-1))) / xRInit;
        }
    }
    else
    {

        if (xR == xS)
        {
            Y = yR + yS;
            N = nR + nS;
            Y = (N * (-1)) / Y;

            X = (((yR * Y) * (-1)) + (nR * (-1))) / xR;
        }
        else
        {
            double xRInit = xR;
            double yRInit = yR;
            double nRInit = nR;

            xR *= (xS);
            yR *= (xS);
            nR *= (xS);

            xS *= xRInit;
            yS *= xRInit;
            nS *= xRInit;

            Y = yR + yS;
            N = nR + nS;
            Y = (N * (-1)) / Y;

            X = (((yRInit * Y) * (-1)) + (nRInit * (-1))) / xRInit;
        }
    }

    ponto.X = X;
    ponto.Y = Y;
    return ponto;
}
```

O método `PontoInterseccaoRetas()` recebe os valores a, b e c de duas equações, retornando um objeto `Point` contendo às coordenadas x e y do ponto de intersecção entre as duas retas.

O terceiro método é responsável por obter a distância entre dois pontos e é apresentado na Figura 29, a seguir.

Figura 29 - Método DistDoisPontos.

```
111     //Encontra a distância entre dois pontos.
112     //Dab = √(xb - xa)² + (yb - ya)²
            9 references
113     public static double DistDoisPontos(double xa, double ya, double xb, double yb)
114     {
115         return Math.Sqrt(Math.Pow((xb - xa), 2) + Math.Pow((yb - ya), 2));
116     }
```

De acordo com a Figura 29 o método `DistDoisPontos()` recebe quatro valores correspondentes às coordenadas x e y de dois pontos, representados por a e b, realizando o cálculo correspondente à equação (2) que recebe as coordenadas bidirecionais de dois pontos e retorna a distância entre eles.

Ao final, foi desenvolvido o método `LeiCosseno90()`, apresentado a seguir na Figura 30.

Figura 30 - Método LeiCosseno90.

```
            1 reference
134     public static double LeiCosseno90(double ab, double ac, double bc)
135     {
136         double cosA = ((bc * bc) + (((ab * ab) + (ac * ac)) * -1)) / (-2 * (ab * ac));
137         if (cosA < 0) cosA = cosA * -1;
138         if (cosA <= 1) return Math.Acos(cosA) * 180 / Math.PI;
139         return -1;
140     }
```

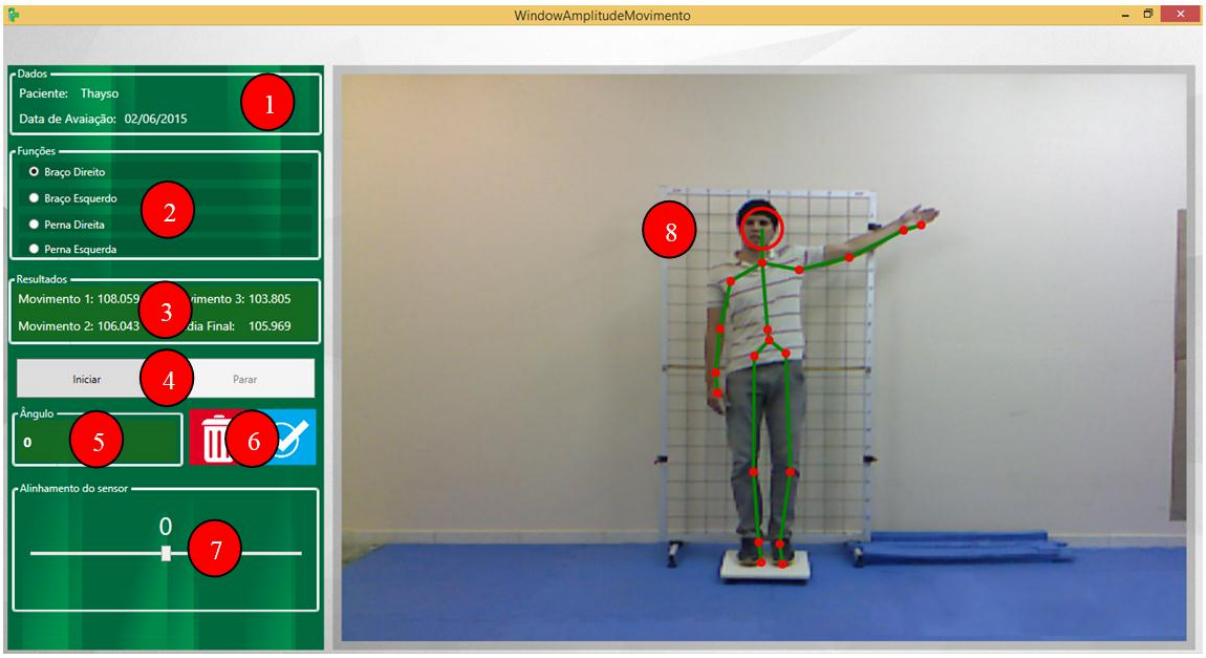
Este método é responsável por utilizar a lei de cossenos para obter um ângulo especifico (ângulo existente entre os lados `ab` e `ac`), usando para isto os três valores correspondentes aos lados de um triângulo. O resultado obtido é dado em radiano, sendo este resultado convertido para um valor positivo correspondente a um ângulo que vai de 0 e 90 graus.

A utilização destes métodos será apresentada na seção seguinte, onde será apresentada a interface e descrito os métodos utilizados no processo de avaliação da ADM.

4.2.5 Avaliação da ADM

Após entrar no módulo de avaliação da Amplitude de Movimento, da inicialização do sensor kinect e habilitação dos fluxos de dados, a ferramenta apresentará a seguinte interface de avaliação, ilustrada pela figura 31.

Figura 31 - Interface de avaliação da ADM.



A Figura 31 apresenta os componentes visuais contidos na interface de avaliação, onde são apresentados os dados do paciente avaliado e a data de avaliação (1). Para que seja realizada a avaliação, primeiramente deve ser selecionado o segmento a ser avaliado no grupo Funções (2) e clicar no botão “Iniciar” para que comece a avaliação, e no botão “Parar” ao termino da avaliação (4).

O processo de avaliação do ângulo deve ser repetido três vezes, sendo os resultados apresentados em (3), assim como uma média dos três movimentos. Durante o processo de avaliação o ângulo atual gerado pelo movimento é apresentado em tempo real (5).

Após a avaliação de ADM do paciente pode ser salvo o resultado da avaliação no banco de dados, clicando no botão salvar de cor azul apresentado em (6), caso ocorra falhas no processo de avaliação, a interface permite o cancelamento da avaliação pelo botão cancelar representado pela cor vermelha.

A interface apresenta em (7) o campo “Alinhamento do Sensor”, que permite a movimentação da base motorizada do sensor em sentido vertical, para que o campo de visão possa ser ajustado e enxergar o paciente durante o processo de avaliação. As imagens obtidas pelo sensor são apresentadas em (8), apresentado o campo de visão do kinect, o paciente e a representação visual do esqueleto capturado pelo sensor.

Para realizar o controle do processo de avaliação da ADM, foi necessário a criação de um objeto do tipo booleano denominado `ativo` sendo inicializado como `false`. Este objeto

é setado como `true` no momento que se inicia a avaliação através do evento click do botão "Iniciar", e setado como `false` por meio do evento click do botão "Parar".

Este objeto é utilizado para realizar o controle do método `CapturarADM()`, que é executado a cada novo *Frame* capturado pelo sensor, fazendo com que o processo de avaliação da ADM só seja executado se o sistema estiver em processo de avaliação, como apresentado pela Figura 32.

Figura 32 - Método CapturarADM.

```
211     private void CapturarADM()
212     {
213         if (ativo == true)
214         {
215             SkeletonFrame esqueleto = null;
216             Skeleton esq;
217             Point baseSegmento = new Point();
218             Point pontaSegmento = new Point();
219             char lado = new char();
220 
221             while (esqueleto == null)
222             {
223                 esqueleto = kinect.SkeletonStream.OpenNextFrame(500);
224             }
225             if (esqueleto != null)
226             {
227                 esq = Extensao.ObterEsqueletoUsuario(esqueleto);
228 
229                 if (esq != null)
230                 {
231                     if (ValidarPostura(esq.Joints[JointType.ShoulderCenter], esq.Joints[JointType.HipRight], esq.Joints[JointType.HipLeft]))
232                     {
233                         ErrorPostura.Visibility = Visibility.Hidden;
234                         Point pescoco = new Point(esq.Joints[JointType.ShoulderCenter].Position.X, esq.Joints[JointType.ShoulderCenter].Position.Y);
235                         Point coluna = new Point(esq.Joints[JointType.Spine].Position.X, esq.Joints[JointType.Spine].Position.Y);
```

O método `CapturarADM()` assim como apresentado pela Figura 32, utiliza o objeto `ativo`, como forma de controle de execução (linha 213), sendo executadas as seguintes linhas do método apenas se o objeto `ativo` estiver setado como `true`, significando que a aplicação se encontra em processo de avaliação.

Nas linhas 215 e 216, é declarada a variável `esqueleto` da classe `SkeletonFrame` e a variável `esq` da classe `Skeleton`, que serão utilizados na manipulação dos esqueleto do paciente durante o processo de avaliação da ADM. Em seguida são declarados dois objetos `Point` (linha 217 e 218), que serão utilizados para guardar as coordenadas dos pontos do segmento avaliado, assim como o objeto `lado`, que irá guardar a informação do lado do segmento avaliado, podendo conter 'D' significando direita, e 'E' para esquerda.

Após declarar as variáveis, foi utilizado um laço de repetição para verificar se o objeto `esqueleto` se encontra vazio ou *null* (linha 221), sendo o objeto instanciado com um novo quadro de dados de esqueleto disponível pelo kinect. A chamada do próximo quadro de dados de esqueleto se dá pelo método `OpenNextFrame()` da classe `SkeletonStream`, sendo passando por parâmetro a quantidade de tempo de espera em milissegundos para retorno da função (linha 223).

Na linha 225, é feita uma nova verificação sobre o objeto `esqueleto`, para que em seguida, linha 227, o objeto `esq` recebesse as informações do esqueleto capturado no *Frame*, sendo utilizado o método `ObterEsqueletoUsuario()` classe `Extensao`, que retorna o primeiro esqueleto rastreado no *Frame*.

Em seguida, verifica se houve algum esqueleto detectado (linha 229) e se o paciente se encontra em uma postura correta na realização do movimento (linha 231). Para validação da postura é utilizado o método `ValidarPostura()`, que verifica se o valor da coordenada x do pescoço se encontra entre as coordenadas x da cintura direita e esquerda, como apresentado pela Figura 33.

Figura 33 - Método ValidarPostura.

```
327         private static bool ValidarPostura(Joint pescoco, Joint quadrilDireito, Joint quadrilEsquerdo)
328         {
329             if (pescoco.Position.X < quadrilDireito.Position.X && pescoco.Position.X > quadrilEsquerdo.Position.X)
330             {
331                 return true;
332             }
333             return false;
334         }
```

Caso o resultado do método `ValidarPostura()` seja `true`, significa que o paciente está com a postura correta, sendo desabilitado o aviso de erro de postura e dado continuidade ao processo de avaliação do ângulo.

Para o processo de avaliação do ângulo se obtém o ponto do pescoço e da coluna, sendo criados dois objetos `Point` e instanciados com as coordenadas x e y das articulações do pescoço "*ShoulderCenter*" (linha 234), e da coluna "*Spine*" (linha 235).

Ainda no método `CapturarADM()`, após a obtenção dos pontos do pescoço e da coluna, é verificado qual segmento está sendo avaliado para instanciar os valores das coordenadas x e y aos pontos da base e ponta do segmento, como ilustrado pela Figura 34.

Figura 34 - Instanciação dos pontos do segmento avaliado.

```
239 baseSegmento = new Point(esq.Joints[JointType.ShoulderRight].Position.X, esq.Joints[JointType.ShoulderRight].Position.Y);
240 pontaSegmento = new Point(esq.Joints[JointType.ElbowRight].Position.X, esq.Joints[JointType.ElbowRight].Position.Y);
241 lado = 'D';
```

Como apresentado pela Figura 34 são instanciados os pontos do segmento avaliado, sendo atribuída na variável `baseSegmento` (linha 239) uma das articulações dos ombros para avaliação do membro superior, ou uma das articulações das virilhas para avaliação de um membro inferior.

Para instanciação da variável `pontaSegmento` (linha 240) utiliza-se uma das articulações dos cotovelos para avaliação do membro superior, ou uma das articulações dos

joelhos para avaliação do membro inferior. Ao final, se atribui na variável `lado` (linha 241) o lado do segmento avaliado.

Assim como apresentado na seção 4.2.3, as articulações têm acesso às informações através da propriedade `Joint`, que são identificadas pela propriedade `JointType`, sendo utilizado na avaliação de cada segmento os *Joints* apresentados pela Tabela 3.

Tabela 3 - Joints por segmento avaliado.

Avaliado	Base	Ponta
Braço direito	*ShoulderRight*	*ElbowRight*
Braço esquerdo	*ShoulderLeft*	*ElbowLeft*
Perna direita	*HipRight*	*KneeRight*
Perna esquerda	*HipLeft*	*KneeLeft*

Após a seleção do segmento avaliado e instanciação dos valores das coordenadas nos pontos das articulações, se realiza a chamada do método `calcularAngulo()`, passando os pontos do pescoço, coluna, base do segmento, ponta do segmento e o do lado avaliado, como ilustrado pela Figura 35.

Figura 35 - Método CalcularAngulo().

```
281   //método calcular Angulo, recebe os pontos corporais e o lado do segmento e retorna o angulo do movimento.
      1 reference
282   private double CalcularAngulo(Point pescoco, Point coluna, Point baseSegmento, Point pontaSegmento, char lado)
283   {
284       List<double> r = new List<double>();
285       List<double> s = new List<double>();
286       Point pontoInterseccao = new Point();
287       double AB = 0, AC = 0, BC = 0;
288
289       r = KinectMath.EquacaoGeralReta(pescoco.X, pescoco.Y, coluna.X, coluna.Y);
290       s = KinectMath.EquacaoGeralReta(baseSegmento.X, baseSegmento.Y, pontaSegmento.X, pontaSegmento.Y);
291
292       pontoInterseccao = KinectMath.PontoInterseccaoRetas(r[0], r[1], r[2], s[0], s[1], s[2]);
293
294       AB = KinectMath.DistDoisPontos(pontoInterseccao.X, pontoInterseccao.Y, baseSegmento.X, baseSegmento.Y);
295       AC = KinectMath.DistDoisPontos(pontoInterseccao.X, pontoInterseccao.Y, pescoco.X, pescoco.Y);
296       BC = KinectMath.DistDoisPontos(baseSegmento.X, baseSegmento.Y, pescoco.X, pescoco.Y);
297
298       return Math.Round(CorrigirAngulo(KinectMath.LeiCosseno90(AB, AC, BC), baseSegmento, pontaSegmento, pontoInterseccao, lado), 3);
299   }
```

Conforme ilustrado pela Figura 35, o método `CalcularAngulo()` é responsável por calcular o ângulo do movimento, assim como foi apresentado pela seção 2.6. Para isto, foram criadas duas listas, a lista ‘`r`’ (linha 284) e lista ‘`s`’ (linha 285) para guardar as informações das duas retas utilizadas no processo de avaliação. Em seguida, foi declarada uma variável `pontoInterseccao` para guardar o ponto de intersecção das duas retas (linha 286), e declaradas as variáveis `AB`, `AC` e `BC`, que serão utilizadas para receber o tamanho dos lados do triângulo (linha 287).

Em seguida, as listas foram instanciadas com o resultado do método `EquacaoGeralReta()`, sendo retornado para a lista ‘`r`’ a equação geral da reta que

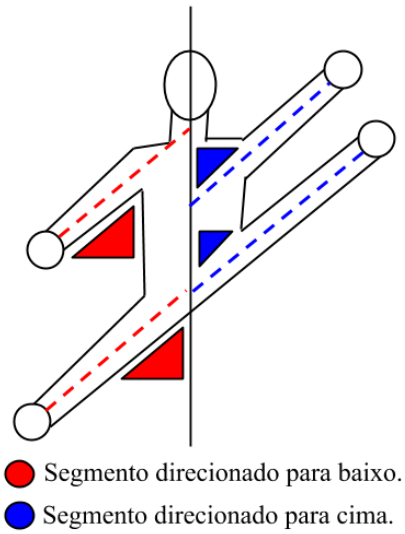
passam pelos pontos do pescoço e coluna (linha 289), e para a lista ‘s’ a equação da reta que passa pelos pontos da base e ponta do segmento avaliado (linha 290).

A variável `pontoIntersecao` recebe na linha 292 o resultado do método `PontoInterseccaoRetas()`, que recebe duas equações e retorna o ponto em que elas se cruzam. Nas linhas seguintes se utiliza o método `DistDoisPontos()` na instanciação da variável `AB` que recebe a distância entre o ponto de intersecção e a base do segmento (linha 294), da variável `AC` com a distância entre o ponto de intersecção e o ponto do pescoço (linha 295), e da variável `BC` com a distância da base do segmento e o pescoço.

Ao final, linha 298, se retorna o resultado do método `LeiCossenos90()`, que recebe os lados de um triangulo e retorna o ângulo de um lado convertido de 0 a 90 graus. O valor retornado pela chamada deste método é passado como parâmetro para o método `CorrigirAngulo()`, juntamente com as variáveis `baseSegmento`, `pontaSegmento`, `pontoInterseccao` e `lado`, sendo o valor do ângulo convertido para três casas decimais após a vírgula pela função `Round` da classe `Math`.

Durante o processo da obtenção de ADM, é utilizado o ponto de intersecção das duas retas, podendo este ponto variar sua posição de acordo com o movimento do paciente. O ponto de intersecção pode estar acima da base do segmento quando o paciente está com o segmente direcionado para baixo, ou estando abaixo da base do segmento quando o paciente está com o segmento direcionado para cima, como ilustrado pela Figura 36.

Figura 36 - Localização do ponto de Intersecção em movimento.



Para identificar e corrigir o valor retornado pelo método `LeiCossenos90()`, foi criado o método `CorrigirAngulo()` apresentado pela Figura 37.

Figura 37 - Método CorrigirAngulo.

```
301  //método de Correção do Angulo, passando o valor y da articulação (cotovelo ou joelho) e o valor y do ponto de interseccao.
     1 reference
302  private double CorrigirAngulo(double anguloAtual, Point baseSegmento, Point pontaSegmento, Point pontoInterseccao, char lado)
303  {
304
305      if ((pontaSegmento.X > baseSegmento.X && lado == 'E') || (pontaSegmento.X < baseSegmento.X && lado == 'D'))
306      {
307          if (pontaSegmento.Y > baseSegmento.Y)
308              return 180;
309          else
310              return 0;
311      }
312      else
313      {
314          if (pontoInterseccao.Y > baseSegmento.Y)
315              return anguloAtual;
316          else
317          {
318              if (pontoInterseccao.Y == baseSegmento.Y)
319              {
320                  return 90;
321              }
322          }
323          return 180 - anguloAtual;
324      }
325  }
```

Como apresentado, o método `CorrigirAngulo()` realiza correções sobre o ângulo obtido pelo método `LeiCossenos90()`, levando em consideração a posição dos pontos da base e ponta do segmento, do ponto de intersecção e o lado do segmento.

Com base na visão do sensor kinect o processo de correção se inicia na linha 305, onde é verificado se o segmento encontra-se cruzado à linha central do corpo. Para isto, é verificado os valores das coordenada ‘x’ da base e da ponta do segmento, assim como o sentido em que o movimento é realizado.

Para que a ferramenta não apresentasse o ângulo gerado pelo lado contrário ao movimento foi usado um condicional na linha 307, que verifica se a coordenada ‘y’ da ponta do segmento está acima da coordenada ‘y’ da base do segmento, o que indica que o movimento está direcionado para cima, fazendo com que o método retorne 180 como limite da angulação (linha 308), ou caso contrário a aplicação retornará 0 como limite (linha 310).

Para movimentos que estão no sentido correto a avaliação, é verificado se a coordenada ‘y’ do ponto de intersecção é maior que a coordenada ‘y’ da base do segmento (linha 314), indicando que o segmento está direcionado para baixo, sendo retornada a variável `anguloAtual` sem alterações (linha 315).

Na linha 318, verifica se os valores das coordenadas ‘y’ do ponto de intersecção e do ponto da base são iguais, significando que o segmento se encontra alinhado horizontalmente e estando com angulação de 90 graus, sendo o valor retornado na linha 320.

Para que o método chegue na linha 323, significa que o paciente encontra-se com o segmento direcionado para cima, sendo assim subtraído da angulação máxima o valor da

angulação do ponto de intersecção, apresentando o ângulo do movimento. O ângulo subtraído pode ser visualizado pelos triângulos azuis apresentados pela Figura 36.

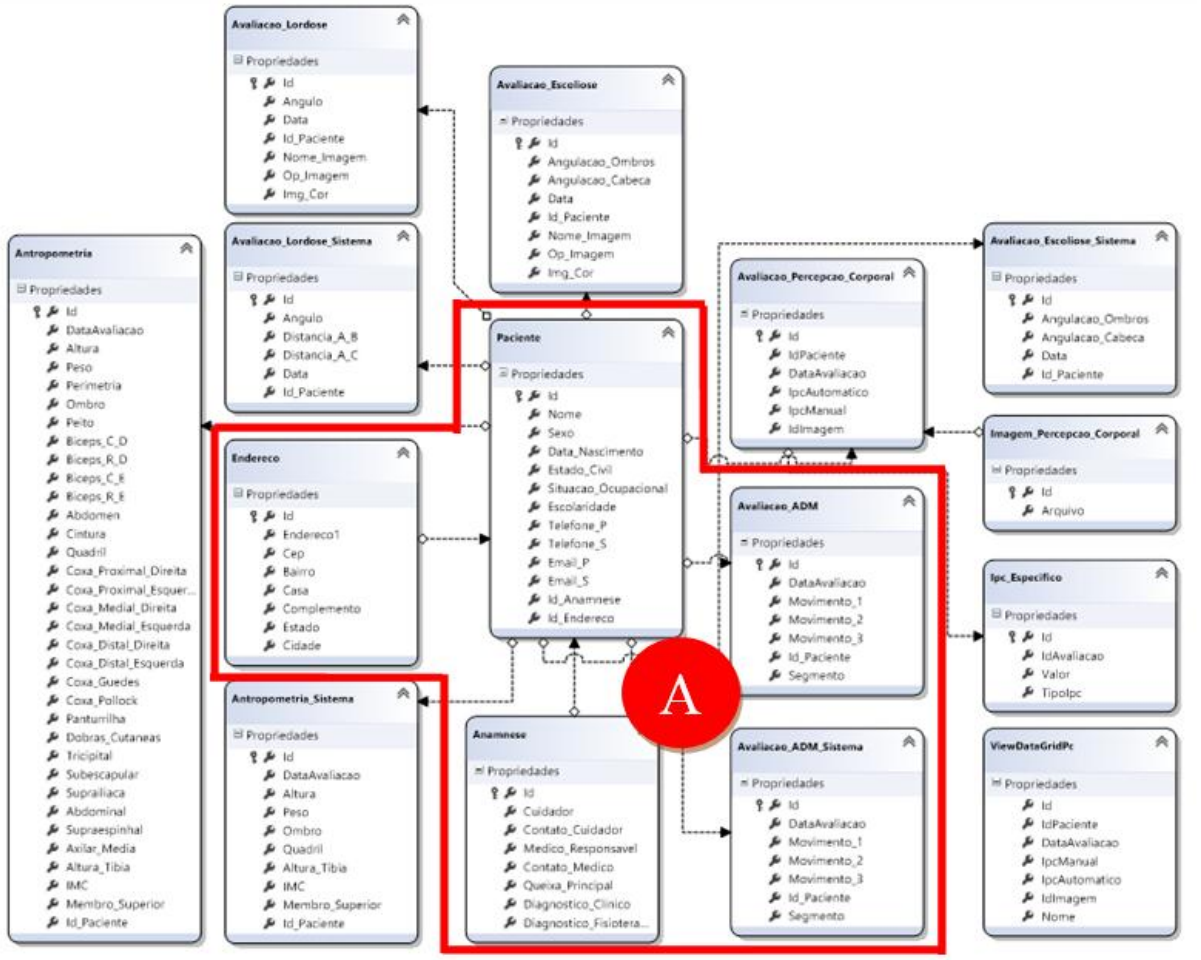
Para realizar a comunicação entre o banco de dados e a aplicação foi utilizado o LINQ to SQL, que conforme a Microsoft (2015, *online*) oferece uma infraestrutura de gerenciamento de dados relacionais como objetos, mapeando os dados de um banco de dados relacional e transformando para um modelo de objeto expresso.

Para organização do método de comunicação entre a aplicação e o banco de dados foi criada a pasta `Manager`, contendo um conjunto de classes utilizadas no processo de inserção, alteração, exclusão e consulta dos dados.

Para armazenamento dos dados obtidos pela aplicação foi necessário a criação do banco de dados, que através das classes e métodos contidos no `Manager` e a utilização do LINQ to SQL, possibilitaram a manipulação das informações dos pacientes e avaliações.

Inicialmente foram criadas as tabelas: `Paciente`, `Endereco`, `Anamnese`, `Avaliacao_ADM` e `Avaliacao_ADM_Sistema`, sendo estas tabelas acopladas ao sistema principal do grupo de pesquisa e fazendo parte de um banco de dados maior. O modelo relacional do banco de dados pode ser visualizado a seguir pela Figura 38.

Figura 38 - Modelo relacional do banco de dados.



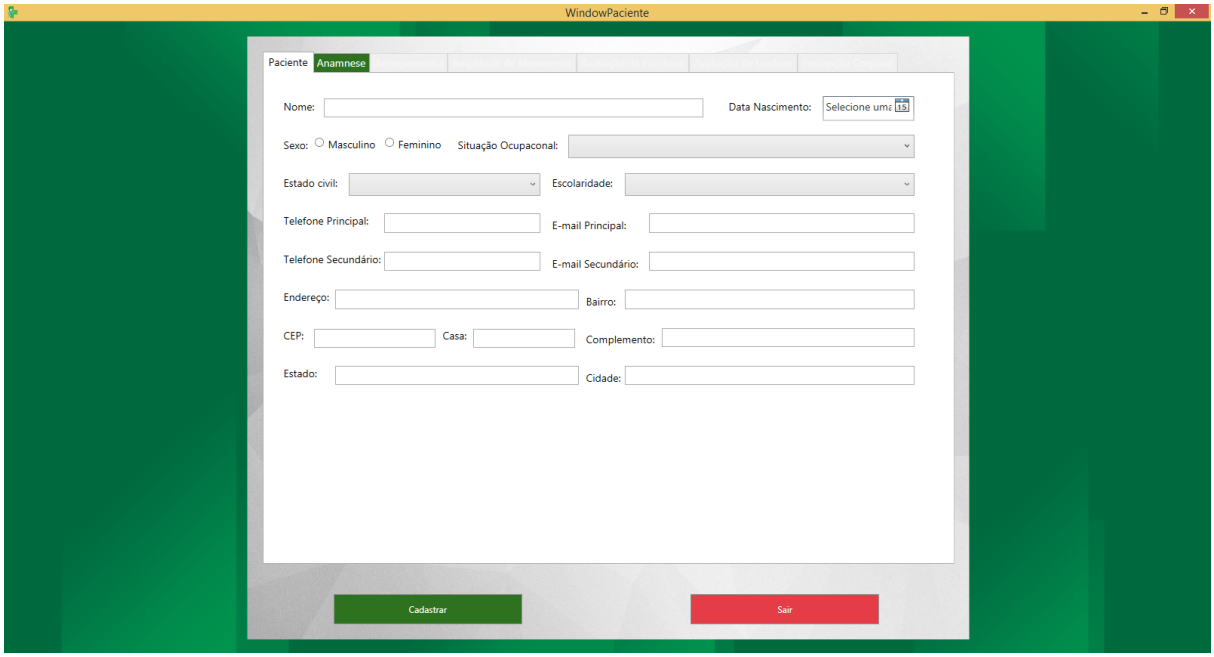
Conforme demonstrado pela Figura 38, as tabelas utilizadas pelo módulo de Amplitude de Movimento são apresentadas em (A), sendo estas as tabelas:

- `Paciente`: na qual possui as informações pessoais do paciente, sendo estas: nome, sexo, data de nascimento, estado civil, situação ocupacional, escolaridade, telefones e e-mails para contato, além de um endereço e uma anamnese.
- `Endereco`: contém as informações referentes ao endereço do paciente, contendo informações do endereço, cep, bairro, casa, complemento, cidade e estado.
- `Anamnese`: esta tabela possui um histórico clínico do paciente, contendo informações relacionadas ao histórico de doenças, exames realizados, uso de medicamentos, alergias, entre outras, sendo muito importantes para profissionais da área da saúde. Nesta tabela são armazenados o nome e telefone do cuidador ou pessoa responsável pelo paciente, o nome e telefone do médico responsável, suas queixas, além dos diagnósticos clínico e fisioterapêutico.

- `Avaliacao_ADM_Sistema`: esta tabela é a responsável por conter todos os dados obtidos no processo de avaliação utilizando o sensor Kinect, sendo armazenados o nome do segmento avaliado, a data da avaliação e os valores dos três movimentos realizados. Para armazenar estes dados se utiliza a interface de avaliação da Amplitude de Movimento (Figura 31).
- `Avaliacao_ADM`: esta tabela contém o nome do segmento, a data da avaliação e o resultado de três movimentos realizados pelo paciente, sendo armazenados os dados obtidos por meio de métodos tradicionais de avaliação, como a goniometria, e inseridos manualmente pelo fisioterapeuta.

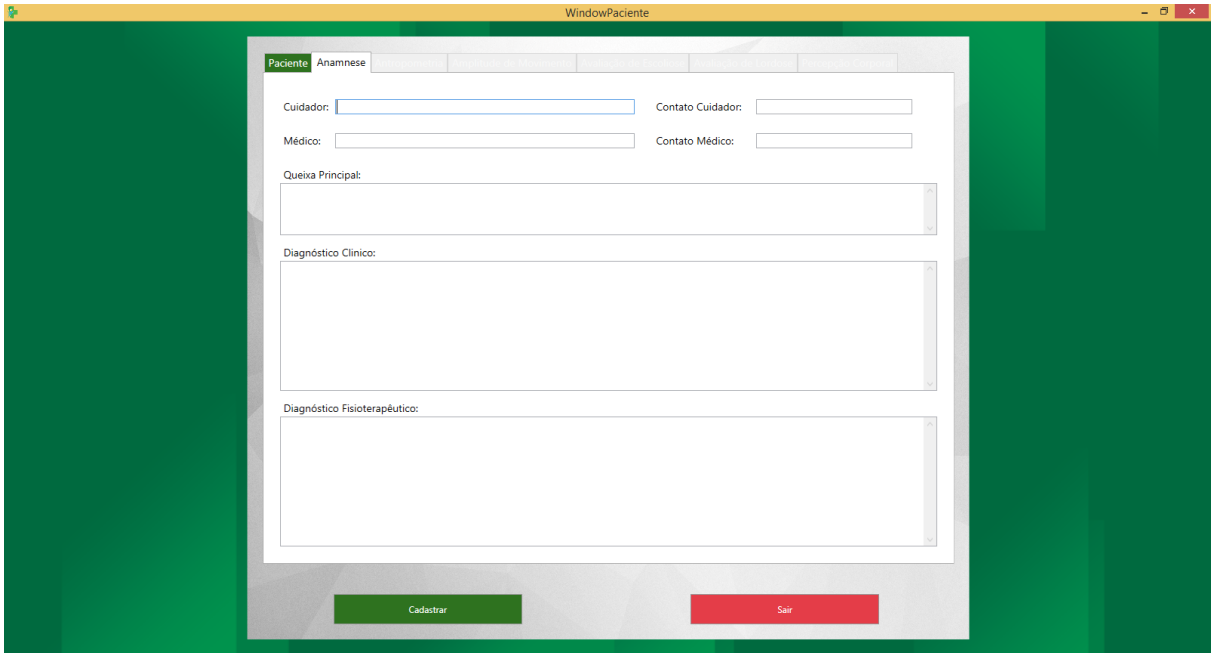
Como apresentado, a aplicação permite o armazenamento de dados, que utilizam as interfaces para inserção de informações. Para inserir dados nas tabelas pacientes, endereço, anamnese e antropometria, é utilizada a interface `WindowPaciente`, que se divide em um grupo de campos através do elemento `TabControlDados`. A `tabPaciente` utilizada para cadastramento de paciente pode ser apresentada pela Figura 39.

Figura 39 - TabPaciente da Interface de cadastro.



Através desta interface o fisioterapeuta poderá realizar o cadastro do paciente, sendo necessário preencher os campos com os dados do paciente na `tabPaciente` apresentada pela Figura 39 e da `tabAnamnese` apresentada a seguir pela Figura 40.

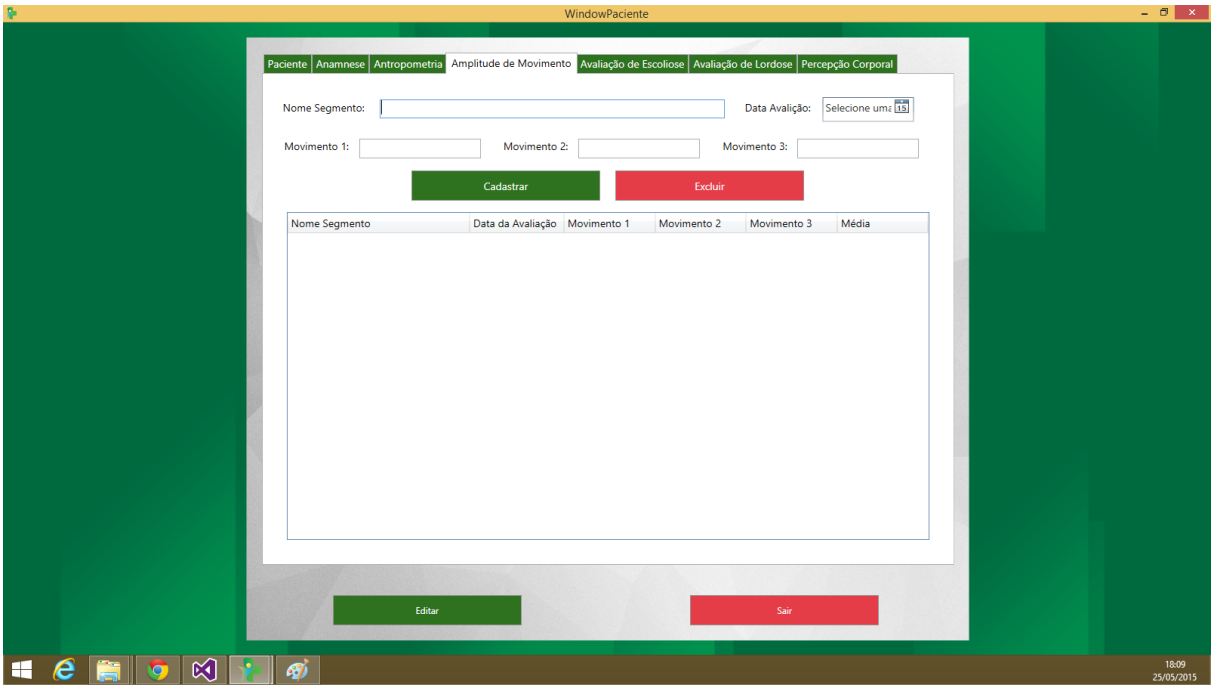
Figura 40 - TabAnamnese da Interface de cadastro.



Após preencher os dados do paciente e realizar o cadastro através do botão `Cadastrar`, os dados serão armazenados nas tabelas `Paciente`, `Endereco` e `Anamnese`, apresentadas anteriormente.

Para permitir o armazenamento das avaliações de ADM manuais obtidos por outros métodos de avaliação, foi criada a `tabADM` apresentada pela Figura 41.

Figura 41 - TabADM da Interface de cadastro.



Por meio da `tabADM`, ilustrada pela Figura 41, é possível realizar o cadastro manual de avaliações de amplitude de movimento, preenchendo os campos e clicando no botão `Cadastrar`. Após realizado o cadastro, os dados são armazenados na tabela `Avaliacao_ADM` e adicionada na lista que contém todas as avaliações de ADM pertinentes ao paciente realizadas de forma manual.

4.3 Testes e Comparação

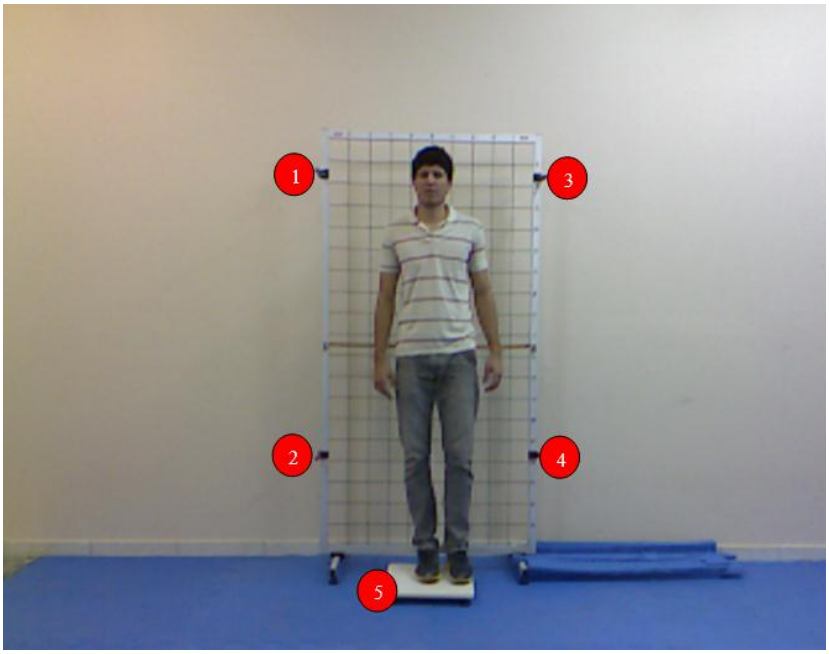
Nesta seção, serão apresentados os testes e comparação da ferramenta, apresentando o ambiente, o método utilizado na coleta dos dados e os resultados obtidos durante a comparação da ferramenta como um método tradicional.

O local utilizado para aplicação das avaliações foi o Laboratório de Tecnologia em Saúde (LTS) do Complexo de Informática do Centro Universitário Luterano de Palmas (CEULP/ULBRA), por se tratar de um ambiente iluminado e sem correntes de ar, sendo situações que podem alterar os dados obtidos pelo sensor.

Devido a necessidade de se obter medidas precisas na obtenção de atributos antropométricos, para que fosse realizado a validação da aplicação, foi preciso identificar o local e o espaço dentro do campo de visão do sensor que proporcionasse o melhor resultado para obtenção dos dados.

O sensor kinect foi posto sobre um balcão de 1,14 metro de altura em relação ao chão. Esta altura está em conformidade com a sugestão do suporte da Xbox (2015, *online*), que indica que o sensor kinect funciona melhor quando posicionado entre 0,6 e 1,8 metros do chão. A distância entre o sensor e a borda do balcão foi de 1,5 centímetros, para que permitisse a visualização dos pés do paciente a ser avaliado e seguir as orientações do suporte da fabricante do sensor. Durante o processo de avaliação o sensor possuía a seguinte visão (Figura 42).

Figura 42 - Local de Avaliação.



Devido a utilização do “modo padrão” (*Default Mode*), que permite um campo de visão entre 0,8m e 4m, a distância de captura utilizada na avaliação foi de 3 metros entre o sensor e o paciente, e a avaliação ocorreu encima de um suporte (5) de 5,5 cm de altura.

Durante o processo de avaliação foram definidos pontos fixos que limitassem o movimento de ADM, fazendo com que os movimentos alcançassem a mesma altura durante as avaliações. Estes pontos são apresentados pela Figura 42, sendo o ponto (1) utilizado para avaliações do braço direito, (2) para avaliações da perna direita, (3) para avaliações do braço esquerdo e (4) para avaliações da perna esquerda.

Os pontos (1) e (3) utilizados no processo de avaliação dos braços foram postos à uma altura de 185 cm do chão, já os pontos utilizados nas avaliações das pernas (2) e (4) se encontravam sob uma altura de 55 cm.

O método de avaliação manual utilizado foi a goniometria, por ser o método mais utilizado atualmente na avaliação da Amplitude de Movimento, sendo avaliado a ADM do braço direito, braço esquerdo, perna direita e perna esquerda. Entretanto, as avaliações manuais não foram realizadas por um fisioterapeuta capacitado, podendo não corresponder aos reais valores de ADM das avaliações realizadas através da goniometria, mas servindo como parâmetro de comparação para um teste de acurácia.

O método de avaliação de ADM utilizando o sistema realizou a captura de três movimentos dos mesmos segmentos, sendo utilizada a média dos movimentos na comparação

com a goniometria. É válido ressaltar que, durante o processo de avaliação utilizando o sistema, contou-se com o método de avaliação postural não existente na goniometria.

Os resultados obtidos nos testes da ferramenta desenvolvida para mensurar a amplitude de movimento utilizando o sensor kinect são apresentados a seguir, através de tabelas e gráficos que ilustram e comparam os dados obtidos utilizando o sistema desenvolvido com os dados obtidos através da goniometria, o desvio e a média de desvio.

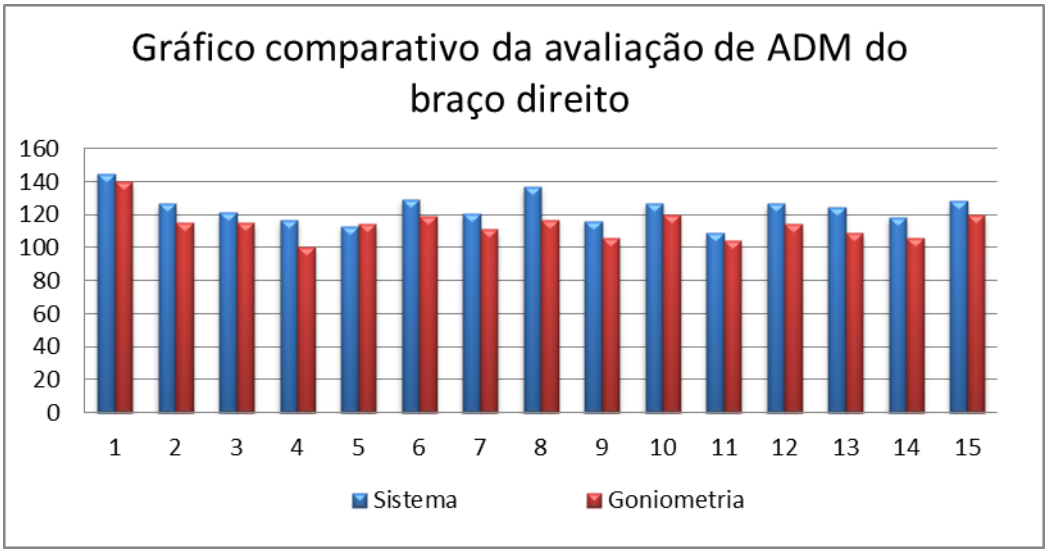
Os dados obtidos na avaliação do ângulo do braço direito são apresentados na Tabela 4.

Tabela 4 - Resultado da avaliação da ADM do braço direito.

Voluntário	**Movimento 1 para ADM do braço direito**	**Movimento 2 para ADM do braço direito**	**Movimento 3 para ADM do braço direito**	**Média dos 3 movimentos para ADM do braço direito**	**Goniometria**	**Diferença entre a Goniometria e a Média do Sistema Kinect**
1	144,2	145,1	145,8	145,1	140	-5,1
2	128,3	128,2	124,0	126,8	115	-11,8
3	120,8	121,4	121,7	121,3	115	-6,3
4	117,6	117,5	115,7	116,9	100	-16,9
5	113,5	112,9	112,9	113,1	114	0,9
6	129,3	128,7	130,1	129,3	119	-10,3
7	120,8	120,8	119,7	120,4	111	-9,4
8	137,8	136,9	136,9	137,2	117	-20,2
9	117,4	116,1	114,8	116,1	106	-10,1
10	129,5	127,8	123,8	127,0	120	-7,0
11	109,7	109,4	106,9	108,7	104	-4,7
12	129,0	125,7	126,9	127,2	114	-13,2
13	124,4	125,0	124,8	124,8	109	-15,8
14	118,2	118,5	118,8	118,5	106	-12,5
15	128,2	127,8	129,5	128,5	120	-8,5
Média ± DP	124,6 ± 9,1	124,1 ± 9,2	123,5 ± 9,8	124,1 ± 9,3	114,0 ± 9,4	-10,1 ± 5,3

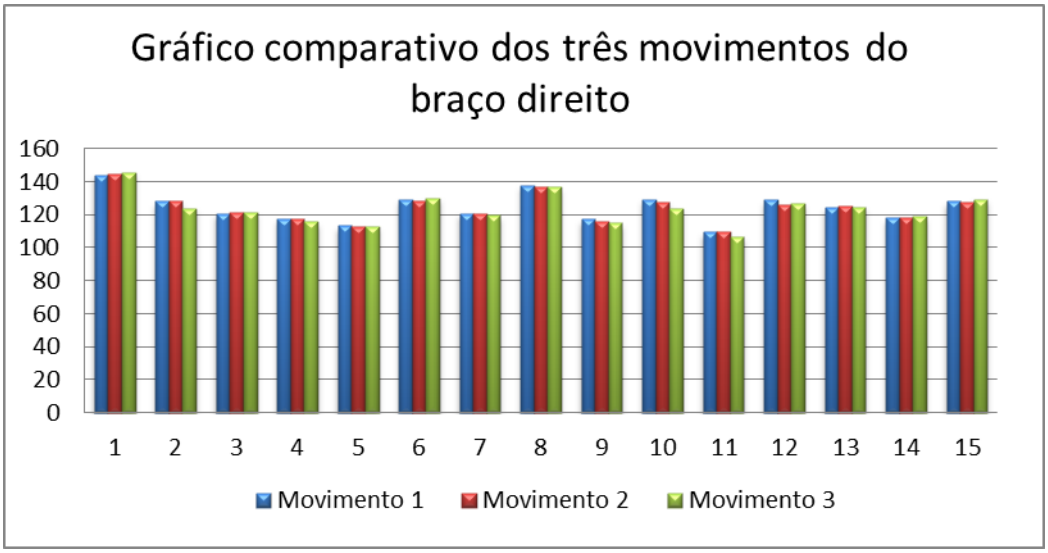
Como apresentado pela Tabela 4, no processo de avaliação utilizando o sistema a média obtida pelos três movimentos apresentou um desvio médio de 10 graus, quando comparado com o resultado obtido através da goniometria. Esta diferença se torna visível através da visualização do gráfico comparativo das avaliações de ADM do braço direito, ilustrado pela Figura 43.

Figura 43 - Gráfico comparativo da avaliação do braço direito.



Porém, quando comparados os três valores dos movimentos avaliados pelo sistema, se apresenta uma proximidade dos resultados, conforme ilustrado pela Figura 44.

Figura 44 - Gráfico comparativo dos três movimentos do braço direito.



Os resultados da avaliação da ADM do braço esquerdo são apresentados a seguir (Tabela 5).

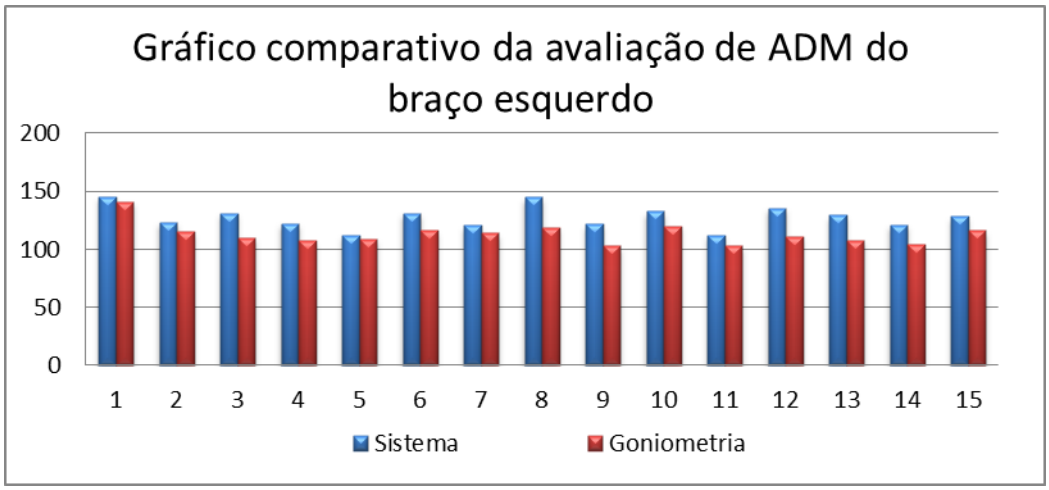
Tabela 5 - Resultado da avaliação da ADM do braço esquerdo.

Voluntário	Movimento 1 para ADM do braço esquerdo	Movimento 2 para ADM do braço esquerdo	Movimento 3 para ADM do braço esquerdo	Média dos 3 movimentos para ADM do braço esquerdo	Goniometria	Diferença entre a Goniometria e a Média do Sistema Kinect
1	146,9	145,3	143,8	145,3	141,0	-4,3
2	124,8	121,3	122,9	123,0	115,0	-8,0

3	129,4	131,5	132,7	131,2	110,0	-21,2
4	123,8	121,0	121,2	122,0	107,0	-15,0
5	112,9	112,0	111,1	112,0	109,0	-3,0
6	129,8	130,7	132,7	131,1	116,0	-15,1
7	120,2	119,4	121,8	120,5	114,0	-6,5
8	144,7	145,2	144,5	144,8	119,0	-25,8
9	122,1	121,1	121,4	121,5	103,0	-18,5
10	131,2	133,4	133,2	132,6	120,0	-12,6
11	113,3	111,6	111,4	112,1	103,0	-9,1
12	136,0	135,3	134,7	135,3	111,0	-24,3
13	128,2	129,2	130,0	129,1	108,0	-21,1
14	119,9	123,0	121,1	121,3	104,0	-17,3
15	128,7	127,6	129,1	128,4	116,0	-12,4
Média ± DP	128,1 ± 9,8	127,5 ± 10,2	129,0 ± 10,0	128,4 ± 9,9	111 ± 9,5	-15,0 ± 7,2

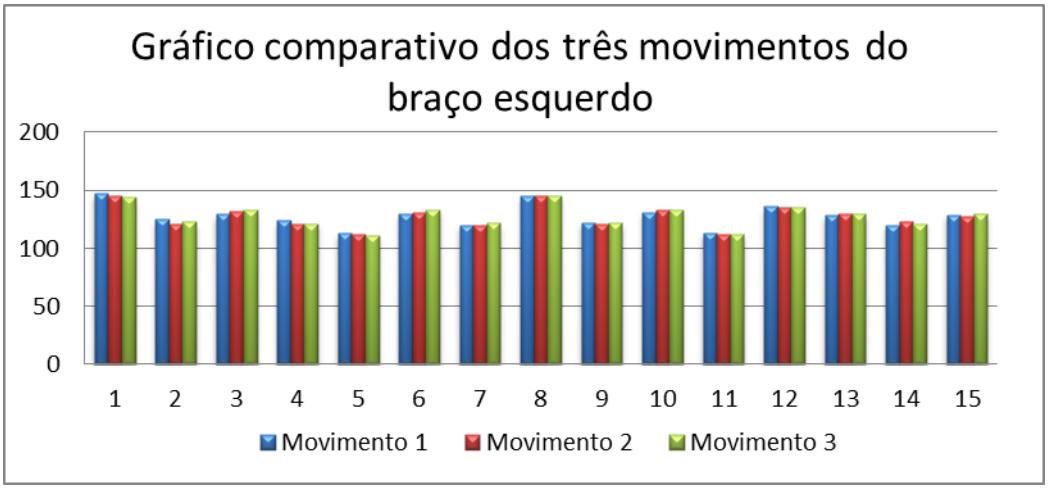
Os resultados das avaliações da amplitude de movimento do braço esquerdo, assim como o resultado das avaliações do braço direito também apresentaram um alto desvio. Em seu pior caso chegando a alcançar uma diferença superior à 25 graus. Porém, sua média de desvio foi de 14,287 graus, sendo uma diferença considerável quando comparado com a goniometria. Esta diferença também pode ser visualizada pelo gráfico da Figura 45.

Figura 45 - Gráfico comparativo da avaliação do braço esquerdo.



Quando comparado os resultados obtidos pela goniometria com os resultados obtidos pela média do sistema se nota uma diferença, porém, quando compara-se os resultados dos três movimentos obtidos pelo sistema os resultados se encontram mais alinhados, apresentando uma igualdade dos resultados do sistema quando comparados com outros valores também obtidos com o sistema, como pode ser visto pelo gráfico da Figura 46.

Figura 46 - Gráfico comparativo dos três movimentos do braço esquerdo.



Após apresentados os resultados das avaliações dos braços, serão apresentados os resultados obtidos no processo de avaliação da ADM das pernas, iniciando pelos resultados das avaliações da perna direita (Tabela 6).

Tabela 6 - Resultado da avaliação da ADM da perna direita.

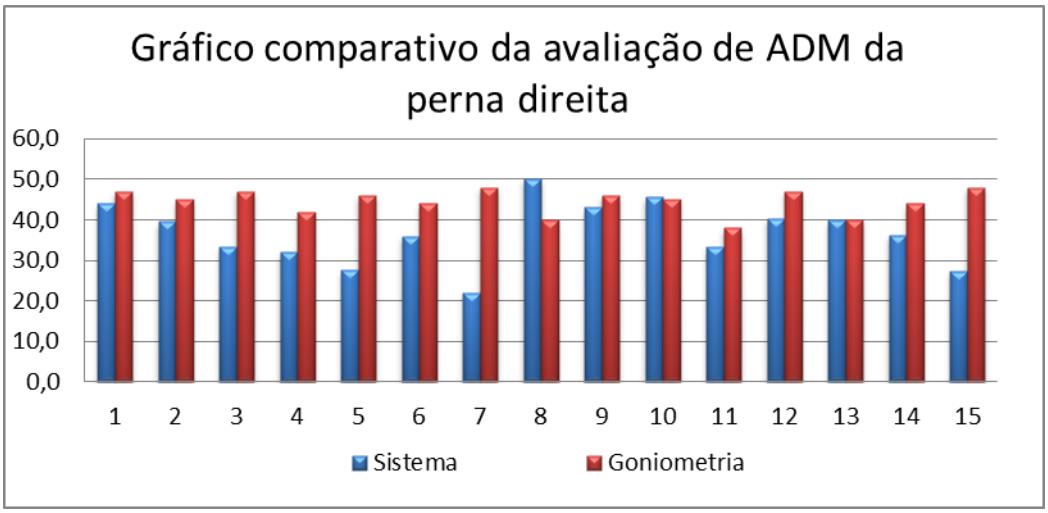
Voluntário	Movimento 1 para ADM da perna direita	Movimento 2 para ADM da perna direita	Movimento 3 para ADM da perna direita	Média dos 3 movimentos para ADM da perna direita	Goniometria	Diferença entre a Goniometria e a Média do Sistema Kinect
1	46,3	42,5	43,5	44,1	47,0	2,9
2	39,2	42,0	37,6	39,6	45,0	5,4
3	35,1	32,3	32,7	33,3	47,0	13,7
4	32,0	31,3	32,6	32,0	42,0	10,0
5	28,0	30,8	23,8	27,5	46,0	18,5
6	36,0	34,3	37,7	36,0	44,0	8,0
7	23,0	21,2	22,0	22,1	48,0	25,9
8	46,5	51,0	53,0	50,2	40,0	-10,2
9	43,3	41,7	44,1	43,0	46,0	3,0
10	46,4	44,9	45,6	45,6	45,0	-0,6
11	33,7	32,6	33,3	33,2	38,0	4,8
12	41,6	40,4	39,1	40,4	47,0	6,6
13	39,1	41,3	39,5	40,0	40,0	0,0
14	36,8	33,8	38,0	36,2	44,0	7,8
15	29,3	26,0	27,1	27,5	48,0	20,5
Média ± DP	36,8 ± 7,1	34,3 ± 7,8	37,7 ± 8,4	36,2 ± 7,6	45,0 ± 3,1	6,6 ± 9,1

Como apresentado pela Tabela 6, os resultados das avaliações da perna direita possuem um desvio médio de 9,197 graus, sendo um desvio considerável, assim como

ocorrido com os resultados dos braços apresentados anteriormente. Porém, este desvio pode ser resultado da utilização do método de validação de postura, não existente na goniometria.

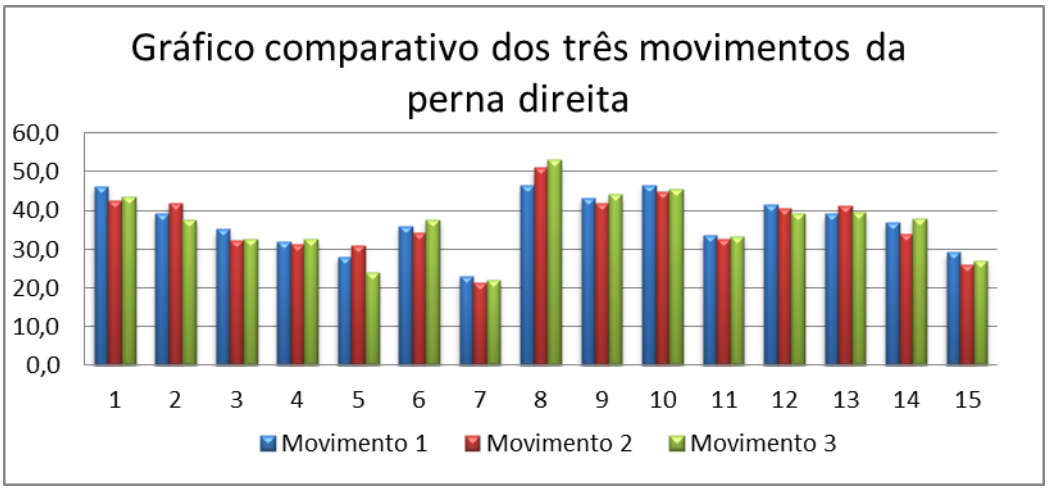
A seguir, Figura 47, é apresentado o gráfico comparativo da avaliação da perna direita, demonstrando uma irregularidade sobre os valores comparados.

Figura 47 - Gráfico comparativo da avaliação da perna direita.



A seguir, Figura 48, será apresentado o gráfico comparativo dos três movimentos obtidos pelo sistema durante a avaliação da perna direita.

Figura 48 - Gráfico comparativo dos três movimentos da perna direita.



Quando comparados os três valores obtidos pelo sistema, percebesse-se que os resultados apresentam uma maior irregularidade se comparado com os gráficos dos braços (Figura 44 e Figura 46). Esta irregularidade se dá devido à dificuldade encontrada por alguns voluntários em alcançar o ponto determinado para realização do movimento da perna, apresentando um desequilíbrio e uma pequena movimentação da perna durante a avaliação.

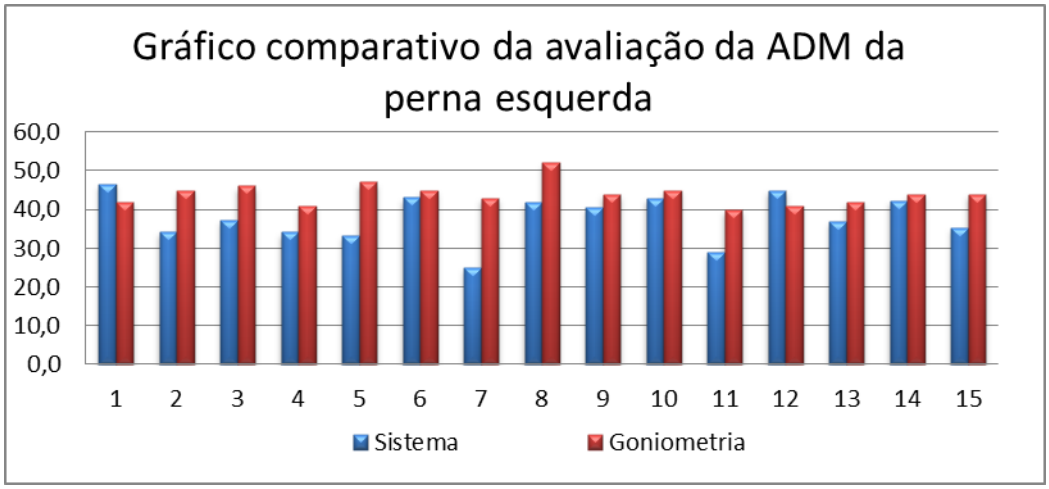
Esta dificuldade também foi enfrentada durante as avaliações da perna esquerda, sendo os resultados apresentados pela Tabela 7.

Tabela 7 - Resultado da avaliação da ADM da perna esquerda.

Voluntário	Movimento 1 para ADM da perna esquerda	Movimento 2 para ADM da perna esquerda	Movimento 3 para ADM da perna esquerda	Média dos 3 movimentos para ADM da perna esquerda	Goniometria	Diferença entre a Goniometria e a Média do Sistema Kinect
1	46,3	42,5	43,5	44,1	47,0	2,9
2	39,2	42,0	37,6	39,6	45,0	5,4
3	35,1	32,3	32,7	33,3	47,0	13,7
4	32,0	31,3	32,6	32,0	42,0	10,0
5	28,0	30,8	23,8	27,5	46,0	18,5
6	36,0	34,3	37,7	36,0	44,0	8,0
7	23,0	21,2	22,0	22,1	48,0	25,9
8	46,5	51,0	53,0	50,2	40,0	-10,2
9	43,3	41,7	44,1	43,0	46,0	3,0
10	46,4	44,9	45,6	45,6	45,0	-0,6
11	33,7	32,6	33,3	33,2	38,0	4,8
12	41,6	40,4	39,1	40,4	47,0	6,6
13	39,1	41,3	39,5	40,0	40,0	0,0
14	36,8	33,8	38,0	36,2	44,0	7,8
15	29,3	26,0	27,1	27,5	48,0	20,5
Média ± DP	36,8 ± 7,1	34,3 ± 7,8	37,7 ±8,4	36,2 ± 7,6	45,0 ± 3,1	6,6 ± 9,1

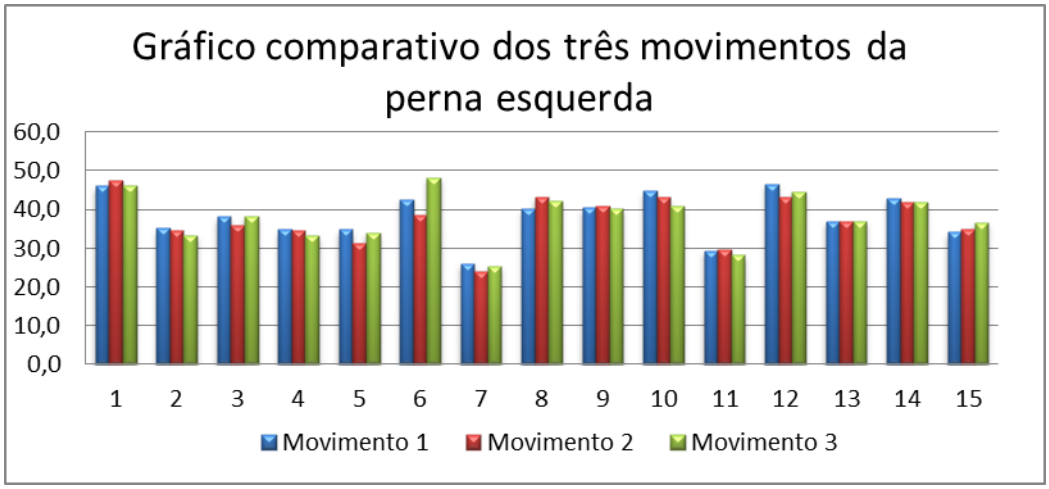
Assim como os resultados das avaliações da perna direita, o desvio médio das avaliações da perna esquerda também possuiu um valor considerável. Os problemas enfrentados foram os mesmos, onde a utilização de uma validação da postura e o desequilíbrio enfrentado por alguns voluntários tornaram os valores desalinhados, como pode ser visualizado através da Figura 49.

Figura 49 - Gráfico comparativo da avaliação da perna esquerda.



Assim como as outras avaliações, foi criado o gráfico comparativos dos três movimentos, sendo o resultado das avaliações da perna esquerda apresentado a seguir (Figura 50).

Figura 50 - Gráfico comparativo dos três movimentos da perna esquerda.



Portanto, nota-se que, para as avaliações dos braços a diferença entre os resultados obtidos através do sistema foram em sua grande maioria maiores que os resultados obtidos através da goniometria. Porém, quando comparados os resultados obtidos pelo sistema, os três movimentos se mantiveram alinhados, mantendo uma proximidade entre os valores capturados.

As avaliações das pernas assim como as avaliações dos braços possuiram uma diferença significativa, porém, os resultados obtidos pelo sistema em sua grande maioria das vezes apresentou valores inferiores aos resultados obtidos através da goniometria.

Quando comparados os valores dos três movimentos realizados pelo sistema, os resultados das avaliações das pernas apresentaram uma diferença superior ao dos braços, sendo estas diferenças motivadas pela dificuldade apresentada por alguns voluntários em alcançar o ponto delimitado (2) e (4), no qual possuia uma altura de 49,5 cm sobre a altura do suporte (5), apresentados pela Figura 42.

Estes resultados apresentam a necessidade de uma metodologia única na avaliação da amplitude de movimento, assim como a necessidade de um profissional capacitado na realização da goniometria, para que os resultados fossem mais aproximados e apresentassem um menor desvio na comparação do sistema com a goniometria.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo implementar e testar uma ferramenta que utilizasse a plataforma *Microsoft Kinect for Windows* e os recursos permitidos pelo *Software Development Kit* (SDK) na realização da captura dos ângulos corporais dos pacientes durante os movimentos de ADM ativo dos ombros e do quadril, encontrados em um plano coronal de visão anterior, e identificar através de testes e comparação com outros métodos de avaliação de ADM se a ferramenta possibilita o fornecimento de dados confiáveis para sua utilização.

Inicialmente, foram realizados estudos sobre os conceitos relacionados ao trabalho, buscando através de referências bibliográficas entender e aprofundar o conhecimento sobre o processo de avaliação da amplitude de movimento, o sensor Microsoft Kinect e seus drivers, sendo contextualizados os conhecimentos na etapa de elaboração do projeto.

Após entender o processo de avaliação da ADM e identificar as articulações disponibilizadas pelo sensor kinect, foram realizados estudos com o intuito de desenvolver métodos de avaliação que através das coordenadas destas articulações permitisse a mensuração dos ângulos da ADM.

Com a criação do método de avaliação da ADM apresentado pela seção 2.6 foi possível iniciar a etapa de implementação da ferramenta, onde foi necessário entender o processo de inicialização do sensor kinect e o funcionamento dos fluxos de cores de imagem, profundidade e rastreamento do esqueleto, assim como o processo de geração dos *Frames* e utilização das articulações do esqueleto.

Após identificar os métodos de obtenção das articulações foi possível comparar os pontos utilizados na avaliação com os pontos disponibilizados pelo fluxo de esqueleto do sensor kinect. Deste modo, foi possível aplicar o método desenvolvido na implementação de um sistema e realizar testes informais na identificação de problemas, sendo criado o método de correção do ângulo e validação da postura. Estes métodos utilizaram os valores das coordenadas das articulações e localização dentro dos eixos do plano cartesiano, para identificar o estado dos segmentos durante a realização do movimento.

Após o desenvolvimento da ferramenta foram realizados testes e comparação com a goniometria, sendo avaliados 15 voluntários com o intuito de identificar se a ferramenta fornece valores úteis na avaliação da amplitude de movimento. Durante os testes observou-se que o sensor pode variar os resultados dependendo a luminosidade e circulação de correntes de ar, sendo necessário a realização da avaliação em ambientes fechados e com boa luminosidade.

Conclui-se, portanto, que os resultados das avaliações quando comparados com a goniometria apresentam diferenças visíveis, mas quando comparado os resultados dos movimentos capturados pela ferramenta se nota uma linearidade, podendo a ferramenta que utiliza o sensor kinect apresentar valores úteis na avaliação da amplitude de movimentos realizados em plano coronal de visão anterior, para movimentos dos ombros e quadris.

Em suma, foi possível entender que o sensor kinect possibilita a avaliação da amplitude de movimento, podendo ser feito novos trabalhos visando obter a ADM de uma maior quantidade de articulações, podendo também elevar o nível de precisão no processo de avaliação da amplitude de movimento através da utilização da nova versão do sensor kinect, que segundo a Xbox (2015, *online*) apresenta melhorias significativas sobre a antiga versão, sendo mais preciso, responsivo, intuitivo e com maior campo de visão.

Além da implementação de novos métodos de obtenção da amplitude de movimento e utilização da nova versão do sensor, se torna necessária a validação ferramenta, para que seja possível a sua utilização no processo de avaliação da amplitude de movimento.

6 REFERÊNCIAS

ALUNOSONLINE. **Equação geral da reta**. Disponível em:< http://www.alunosonline.com.br/matematica/equacao-geral-reta.html>. Acesso em 11 dez. 2014.

AVANCINI, Mattia. **Using Kinect to emulate an Interactive Whiteboard:** Uso del Kinect per emulare lefunzionalità di una Lavagna Interattiva Multimediale. 2011. 113 f. Tese (Doutorado) - Curso de Tecnologia da Informação, Departamento de Department Of Information Engineering And Computer Science, University Of Trento, Trento, 2012. Disponível em: <http://latemar.science.unitn.it/segue_userFiles/LITSA/Avancini_Mattia_138793_Thesis.pdf >. Acesso em: 08 out. 2014.

BALLARD, D. H; BROWN, C. M. **Computer Vision.** New Jersey: Prentice-hall, Inc, 1982. 538 p. Disponível em: <http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/Ballard__D._and_Brown__C._M.__1982__Computer_Vision.pdf>. Acesso em: 8 nov. 2014.

BATISTA, Caroline dos Anjos Bezerra; MEIRA, Mírian Guiar Caires Vasconcelos; SANTANA, Levy Aniceto. Estudo comparativo entre as medidas de goniometria e da fleximetria passiva na articulação do joelho. **Fisioterapia Brasil**, v.11, n. 1, mar/abr. 2010. Disponível em: <http://www.faculdadeguararapes.edu.br/site/downloads/Fisioterapia_mar a abr 2010.pdf>. Acesso em: 21 out. 2014.

BRUNNER, Simon. **USING MICROSOFT KINECT SENSOR TO PERFORM COMMANDS ON VIRTUAL OBJECTS.** 2012. 101 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade de Friburgo, Friburgo, 2012. Disponível em: <https://diuf.unifr.ch/main/diva/sites/diuf.unifr.ch.main.diva/files/Thesis paper.pdf>. Acesso em: 07 out. 2014.

CAMPOS, Guilherme Pires. **Sistema para fisioterapia baseado na plataforma Kinect.** 2012. 70 f. Dissertação (Mestrado) - Curso de Engenharia Eletrotécnica e de Computadores, Universidade do Porto, Porto, 2013. Disponível em: <http://repositorio-aberto.up.pt/handle/10216/67745>. Acesso em: 11 out. 2014.

CARDOSO, Gabriel Schade. **Microsoft Kinect:** Criando Aplicações Interativas com o Microsoft Kinect. São Paulo: Casa do Código, 2013. 181 p.

CARDOSO, Gabriel Schade; SCHMIDT, Ana Elisa Ferreira. Biblioteca de Funções para Utilização do Kinect em Jogos Eletrônicos e Aplicações NUI. In: XI SBGAMES, 4, 2012, Santa Catarina. **Proceedings of SBGames 2012.** Brasília: Feevale, 2012. p. 29 - 32. Disponível em: <http://sbgames.org/sbgames2012/proceedings/papers/computacao/comp-short_08.pdf>. Acesso em: 07 out. 2014.

CASTRO, Vanilde de et al. Variabilidade na aferição de medidas antropométricas: comparação de dois métodos estatísticos para avaliar a calibração de entrevistadores. **Revista Brasileira de Epidemiologia,** São Paulo, v. 11, n. 2, p.278-286, jun. 2008. Disponível em:

<http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1415-790X2008000200009>. Acesso em: 8 nov. 2014.

CATUHE, David. **Programming with the Kinect for Windows Software Development Kit.** Washington: Microsoft Press, 2012. 224 p. Disponível em: <http://filepi.com/i/HMUop4Y>. Acesso em: 11 out. 2014.

CHERCHIGLIA, Letícia Lana. **DESENVOLVIMENTO DE JOGO EDUCATIVO PARA MESA INTERATIVA E KINECT ATRAVÉS DO USO DE VISÃO COMPUTACIONAL.**2011. 22 f. Monografia (Especialização) - Curso de Ciência da Computação, Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Minas Gerais, 2011. Disponível em: <https://www.ufmg.br/frmfa/wp-content/uploads/2012/07/Monografia-POC-1-Letícia-Lana-Cherchiglia.pdf>. Acesso em: 9 out. 2014.

CORREIA, Pedro Pezarat. **Estudo do Movimento: Exercício e Estudos Práticos.** Lisboa: Fmh, 2012. Disponível em: <http://www.fmh.utl.pt/indices/estudomovimento.pdf>. Acesso em: 07 nov. 2014.

COUTO, Natélia Ellery Ribeiro. **SOFTWARE DE ARMAZENAMENTO DE POSES CAPTURADAS PELO MICROSOFT KINECT.** 2012. 88 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí, 2012. Disponível em: <http://siaibib01.univali.br/pdf/Natalia Ellery Ribeito Couto.pdf>. Acesso em: 16 out. 2014.

FILHO, Albertino Oliveira et al. Variabilidade intra-avaliador e inter-avaliadores de medidas antropométricas. **Acta Scientiarum Health Sciences: Sistema de Informação Ciêntifica**, Maringá, v. 29, n. 1, p.1-5, jul. 2007. Semestral. Disponível em: <http://www.redalyc.org/pdf/3072/307226620001.pdf>. Acesso em: 8 nov. 2014.

HALL, Carrie M; BRODY, Lori Thein. **Exercício Terapêutico:** Na Busca da Função. Rio de Janeiro: Guanabara Koogan, 2001. 714 p.

JANA, Abhijit. **Kinect for Windows SDK Programming Guide**: Build motion-sensing applications with Microsoft's Kinect for Windows SDK quickly and easily. Birmingham: Packt Publishing, 2012. 366 p. (978-1-84969-238-0). Disponível em: <http://lirec.ict.pwr.wroc.pl/~jkedzier/urbi/Kinect%20for%20Windows%20SDK%20Programming%20Guide.pdf>. Acesso em: 9 out. 2015.

JUNGONG, Han et al. Enhanced computer vision with microsoft kinect sensor: A review. **IEEE Transactions on cybernetics**, v.43, n. 5, p. 1318 – 1334, Disponível em: <http://lshao.staff.shef.ac.uk/pub/KinectReview_TC2013.pdf>. Acessado em: 07 nov. 2014.

KENDALL, Florence Peterson; MCCREARY, Elizabeth Kendall; POVANCE, Patricia Geise. **Músculos Provas e Funções:** com Postura e Dor. 4. ed. São Paulo: Manole, 1995. 456 p. Tradução de: Lilia Bretrntz Ribeiro.

KINECT FOR WINDOWS. **Skeletal Tracking.** 2014. Developer Network da Microsoft. Disponível em: <http://msdn.microsoft.com/en-us/library/hh973074#ID4ENB>. Acesso em: 11 out. 2014

KISNER, C.; COLBY L. A**. Exercícios Terapêuticos: Fundamentos e Técnicas**. 3. Ed. Manole. 1998. 746 p.

KISNER, C.; COLBY L. A. **Exercícios Terapêuticos: Fundamentos e Técnicas**. 5. Ed. Manole. 2009. 972 p.

KISS, Maria Augusta P D M. **Esporte e exercício: avaliação e prescrição.** São Paulo: Roca, 2003.

MARQUES, Amélia Pasqual. **Manual de Goniometria.** Barueri: Manole 2. Ed. 2003. 81 p.

MATHEWS, D. K. **Medida de avaliação em Educação Física.** 5. ed. Rio de Janeiro: Interamericana, 1980.

MATTOS, Ewerton Luis de. **Utilização de Biofeedback Cinemático com Sensor Kinect em Sessões de Fisioterapia.** 2012. 106 f. Monografia (Especialização) - Curso de Ciência da Computação, Centro de Ciências Tecnológicas da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí (sc), 2012. Disponível em: <http://siaibib01.univali.br/pdf/Ewerton Luis de Mattos.pdf>. Acesso em: 9 out. 2014.

MICROSOFT. **Kinect SDK features**. Disponível em: <http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx>.
Acesso em: 14 out. 2014.

MICROSOFT. Skeletal Tracking. Disponível em: < http://msdn.microsoft.com/en-us/library/hh973074>. Acesso em: 10 dez. 2014.

MOORE, Keith L.; DALLEY Arthur F. **Anatomia Orientada para Clínica**. 4. Ed. Rio de Janeiro: Guanabara Koogan 1999. 1024 p.

NETO, Elias; BRUNO, Odemir. Reconhecimento de gestos em imagens de profundidade utilizando kinect. In: WORKSHOP DE VISÃO COMPUTACIONAL, 8, 2012, Goiânia, **Anais eletrônicos**... Goiânia: UFG, 2012. p. 1-6. Disponível em: <http://iris.sel.eesc.usp.br/wvc/Anais_WVC2012/default.htm>. Acesso em: 08 nov. 2014.

NEUMANN, D. A. **CINESIOLOGIA DO APARELHO MUSCULOESQUELÉTICO: Fundamentos para Reabilitação**. 2. ed. Londres: Elsevier, 2011. 760 p. Tradução autorizada do idioma inglês da edição publicada por Mosby. Disponível em: <http://blogelseviersaude.elsevier.com.br/wp-content/uploads/2012/09/2011-neumann-esample.pdf>. Acesso em: 7 nov. 2014.

NOBESCHI, Leandro. **Introdução ao Estudo da Anatomia**, Virtual Books, 2010. Disponível em: <http://www.imagingonline.com.br>. Acesso em: 13 set. 2014.

PAULA, Bruno Campagnolo de. Adaptando e desenvolvendo jogos para uso com o Microsoft Kinect, 2011, Salvador. **Anais do X Símpósio Brasileiro de Games e Entretenimento Digital.**Salvador: Sociedade Brasileira de Computação, 2011. Disponível em: <http://www.sbgames.org/sbgames2011/proceedings/sbgames/papers/tut/1-kinect_FAAST_Final_MesmoComColunas.pdf>. Acesso em: 14 out. 2014.

PEREIRA, Maurício Gomes. **Epdemiologia: teoria e prática.** Rio de Janeiro: Guanabara Koogan, 1995. 620 p.

QUEIROZ, Murilo. **Um cientista explica o Microsoft Kinect.** 2010. Disponível em: <http://blog.vettalabs.com/2010/10/29/um-cientista-explica-o-microsoft-kinetic/>. Acesso em: 08 out. 2014.

REIS, Vitor Alexandre dos; JORGE, Lúcio André de Castro. RECONSTRUÇÃO DIGITAL DE OBJETOS EM 3D POR NUVEM DE PONTOS UTILIZANDO O KINECT. **Rev Cinentífica Eletrônica Uniseb,** Ribeirão Preto, v. 1, n. 2, p.176-191, dez. 2013. Disponível em: <http://uniseb.com.br/presencial/revistacientifica/arquivos/jul-15.pdf>. Acesso em: 12 out. 2014.

SAMUELR. Visão geral do Kinect e do Kinect SDK. Disponível em: < http://www.samuelr.com/>. Acesso em 16 out. 2014.

SANTOS, Jean Douglas Moura et al. Confiabilidade inter e intraexaminadores nas mensurações angulares por fotogrametria digital e goniometria. **Revista Fisioterapia em Movimento,** Curitiba, v. 24, n. 3, p.389-400, jul. 2011. Trimestral. Disponível em: <http://www.scielo.br/pdf/fm/v24n3/03.pdf>. Acesso em: 8 nov. 2014.

SHOTTON, Jamie et al. Real-time human pose recognition in parts from single depth images. **Communications of the ACM**, v. 56, n. 1, p. 116-124, 2013.

SILVA, Rodrigo Luis Ferreira da et al. Correlação entre fleximetria e goniometria radiológica para avaliações da amplitude articular estática do cotovelo. **Fisioterapia Brasil,** São Carlos, v. 5, n. 12, p.359-364, ago. 2011. Bimestral. Disponível em: <http://www.redeuromh.com/arquivos/estudo previo/rodrigo/FB 2011 Fleximetria Goniometria.pdf>. Acesso em: 10 nov. 2014.

SILVEIRA, Marcus Almeida da. **Técnicas de Navegação em Documentos Utilizando Microsoft Kinect.** 2011. 36 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2011. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/36884/000819161.pdf?seque>. Acesso em: 07 out. 2014.

TAVARES, João Manuel Ribeiro da Silva. **Análise de Movimento de Corpos Deformáveis usando Visão Computacional.** 2000. 323 f. Tese (Doutorado) - Curso de Engenharia, Universidade do Porto, Porto, 2000. Disponível em: <http://paginas.fe.up.pt/~tavares/downloads/publications/teses/tesePhD_JT.pdf>. Acesso em: 8 nov. 2014.

VÁZQUEZ, Francisco Javier. **Brazo Robótico Controlado Mediante Sensor Kinect.** 2013. 138 f. Tese (Doutorado) - Curso de Engenharia Mecânica e Elétrica, Departamento de Instituto Politécnico Nacional, Escola Superior de Engenharia Mecânica e Elétrica, México, 2013.

WEBB, Jarett; ASHLEY, James; **Beginning Kinect Programming with the Microsoft Kinect SDK**. 1. ed. Apress, 2012.

XBOX, Support. **Mais sobre posicionamento do sensor Kinect.** 2015. Microsoft. Disponível em: <http://support.xbox.com/pt-BR/xbox-360/kinect/sensor-placement>. Acesso em: 3 jun. 2015.

YAMADA, Fernando Akio Araújo et al. Reconstrução de Objetos 3D utilizando Estruturas de Indexação Espacial com o Microsoft Kinect, WRVA - **Workshop de Realidade Virtual e Aumentada** - Paranavaí (PR) - Brasil - 2012. Disponível em: < http://www.lbd.dcc.ufmg.br/colecoes/wrva/2012/0023.pdf>. Acesso em 10 out 2014.

ZANDONÁ, Rafael; TANAKA, Sergio Akio, **Estudo da utilização do sensor de movimento (Kinect) na interação entre homem e computador**. In: VI semana Tecnológica dos cursos de Ciência da Computação e Sistema de Informação da UniFil: Governança corporativa em ti e E-Commerce, 2012 Disponível em: <http://www.unifil.br/portal/arquivos/publicacoes/paginas/2012/11/516_865_publipg.pdf > Acesso em 07 out. 2014.