UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS - EESC

STENIO AGUIAR COSTA E SILVA

Sistema de rastreamento de um objeto no espaço através do Kinect e um sensor inercial

São carlos
2016

STENIO AGUIAR COSTA E SILVA

Sistema de rastreamento de um objeto no espaço através do Kinect e um sensor inercial

Trabalho de conclusão de curso apresentado à Escola de Engenharia de São Carlos, Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro de Computação.

Área de Concentração: Dinâmica de máquinas e sistemas

Orientador: Prof. Dr. Glauco Augusto de Paula Caurin.

São Carlos

2016

Aguiar Costa e Silva, Stenio
A282s Sistema de rastreamento de um objeto no espaço através do Kinect e um sensor inercial / Stenio Aguiar Costa e Silva; orientador Glauco Augusto de Paula Caurin. São Carlos, 2016.

Monografia (Graduação em Engenharia de Computação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. visão-computacional. 2. reabilitação. 3. sensor inercial. I. Título.

FOLHA DE APROVAÇÃO

Nome: Stenio Aguiar Costa e Silva

Título: "Sistema de rastreamento de um objeto no espaço através do Kinect e um sensor inercial"

Trabalho de Conclusão de Curso defendido em 21 / 06 / 2016.

Comissão Julgadora:	**Resultado:**
Prof. Associado Glauco Augusto de Paula Caurin (Orientador) - SEM/EESC/USP	Aprovado
Prof. Dr. Valdir Grassi Junior SEL/EESC/USP	Aprovado
Prof. Dr. Leonardo Marquez Pedro UFSCar	Aprovado

Coordenador do Curso Interunidades Engenharia de Computação pela EESC:

Prof. Dr. Maximiliam Luppe

STENIO AGUIAR COSTA E SILVA

DEDICATÓRIA

Dedico este trabalho aos meus pais, Bel e Cléo. Sua paciência e seu suporte são a base para tudo o que posso vir a conquistar.

AGRADECIMENTOS

Agradeço à minha namorada, Rachel Bergantin, pelos momento de tranquilidade e calma que me fornece. A sua companhia e a possibilidade de vislumbrar um futuro com você, me fazem querer me superar em todas as atividades.

À toda minha família por acreditarem em mim e me ajudarem a sempre seguir pelo caminho certo.

Ao meu orientador direto, Gustavo Lahr, por ter confiado e me auxiliado durante todo o desenvolvimento deste trabalho. Por possuir a atenção e dedicação necessários a todos bons professores.

À todos os integrantes do laboratório de mecatrônica da Usp - São Carlos, em especial ao meu orientador Prof. Dr. Glauco Caurin, que sempre prestativos e receptivos, foram parte essencial do desenvolvimento deste trabalho.

Aos funcionários da faculdade representados pela Sra. Silvana Wick, que sempre lutando em favor dos interesses dos alunos, fez com que seja possível me preocupar somente com a graduação.

Aos colegas da república Sparta, república 29 e ao time de handebol Caaso, que nossa amizade possa durar por muitos anos e que tenhamos muitas histórias para compartilhar.

RESUMO

SILVA, S.A.C. Sistema de rastreamento de um objeto no espaço através do Kinect e um sensor inercial - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

A reabilitação de pacientes em recuperação de lesões motoras vem contando com auxílio da robótica e recebendo novas motivações para os pacientes, como o uso de jogos desenvolvidos para fisioterapia. Para acompanhar o tratamento, é interessante rastrear as posições no espaço dos membros em tratamento. Este trabalho sugere uma forma de rastreamento de objetos baseado em sua cor através de um sensor inercial, um Kinect e uma máquina de estados responsável pela tomada de decisões. Os testes foram realizados com uma IMU posicionada no braço de um suposto paciente, enquanto este segurava um objeto vermelho que foi rastreado pelo Kinect. Foram elaborados programas, utilizando MATLAB, para obtenção e tratamento de imagens do Kinect e Visual Studio, para aquisição de dados da IMU. A integração das duas tecnologias permite que o rastreamento de objetos seja efetivo mesmo quando ocorre oclusão do corpo observado, utilizando uma máquina de estados para decidir de qual dos dispositivos o posicionamento deve ser obtido. As comparações dos dados dos erros entre as posições oferecidas pelos dois sensores chegaram a valores menores que 100 mm, o que mostra que a tecnologia pode ser utilizada, com possibilidades de melhoria se for utilizada em ambientes com melhor controle de luz e com a utilização de duas IMU's.

Palavras-chave: visão computacional, reabilitação, sensor inercial.

ABSTRACT

SILVA, S.A.C. Object spacial tracking system using Kinect and inertial sensor – School of Engineering of Sao Carlos, University of Sao Paulo, Sao Carlos, 2016.

The rehabilitation patients recovering from injuries that compromise movement has been relying on the aid of robotics and getting new motivations for patients, such as the use of games developed for physical therapy. To achieve an effective treatment, it is interesting to track the positions in space of the limbs being treated. This work proposes a method of tracking objects based on their color using an inertial measurement unit, Kinect and a state machine responsible for making decisions. The tests were performed with an IMU positioned on the arm of a patient, while he held a red object that was tracked by Kinect. Programs were developed using MATLAB for obtaining and handling Kinect images and Visual Studio, for the acquisition of data from the IMU. The integration of the two technologies allows effective tracking objects even when there is occlusion of the observed body using a state machine to decide from which device the position should be obtained. Comparisons of data errors between positions offered by both sensors were below 100 mm, which shows that the technology can be used, with improvements being possible if it is applied in environments with better control of light and if two IMU's are used together.

Keywords: computational vision, rehabilitation, inertial sensor.

LISTA DE FIGURAS

LISTA DE SIGLAS

USB	*Universal Serial Bus*
UDP	*User Datagram Protocol*
IMU	*Inertial Measurement Unit*
RGB	*Red Green Blue*
DoF	*Degrees of Freedom*

LISTA DE SIMBOLOS

qn	Estado n de uma máquina de estados
NaN	*Not a Number* – valor numérico impossível de ser representado
L	Comprimento do braço do paciente
A	*Yaw*
B	*Pitch*
C	*Row*
FMA	*Fugl-Meyer Assessment*
WMF	*Wolf Motor Function Test*
SIS	*Stroke Impact Scale*

SUMÁRIO

1. INTRODUÇÃO

Depois de acidentes ou alguma doença, é possível que pacientes percam parte da movimentação em membros superiores ou inferiores, sendo necessária sua submissão a atividades de reabilitação. A fisioterapia tradicional fornece bons resultados a estes pacientes, mas em casos mais críticos, onde o paciente pode passar anos fazendo a terapia, é possível que esta se torne uma tarefa maçante e tediosa, o que pode prejudicar no tratamento.

Tentando mitigar esses efeitos colaterais, terapias com jogos e a utilização de diversos equipamentos eletromecânicos, como robôs, auxiliam na recuperação de pacientes em situações de reabilitação pós traumas (KREBS; CAURIN; BATTISTELLA, 2014). Segundo Lo et al. (2010), a reabilitação robótica sozinha em relação à reabilitação tradicional possui uma maior intenção de manter os ganhos em testes ao longo do tempo.

Os gráficos na Figura 1 mostram um comparativo entre a terapia convencional, a assistida por robôs e a terapia intensiva (ICT). Os critérios para comparação são *Fugl-Meyer Assessment* (FMA), *Wolf Motor Function Test* (WMF) e *Stroke Impact Scale* (SIS), todos são índices utilizados para medir o comprometimento motor após uma lesão e durante o tratamento. O índice FMA para o comparativo do tratamento convencional e a assistida por robôs mostra, na Figura 1-A, que a assistida desempenha um papel ao longo do tempo com uma recuperação mais significativa. Comparada com a intensiva, o desempenho da assistida com a intensiva assemelham-se muito, com o contraponto de que a terapia robótica desgasta muito pouco o terapeuta.

Neste contexto, este trabalho visa desenvolver técnicas de mapeamento e reconhecimento de um ponto no espaço que esteja atrelado ao braço de um paciente. É empregado um sistema de visão computacional, entretanto, caso este venha a sofrer algum tipo de oclusão, utiliza-se um sensor inercial e uma máquina de estados para continuar obtendo a posição do paciente. Os capítulos que seguem são responsáveis por apresentar as ferramentas utilizadas e suas implementações.

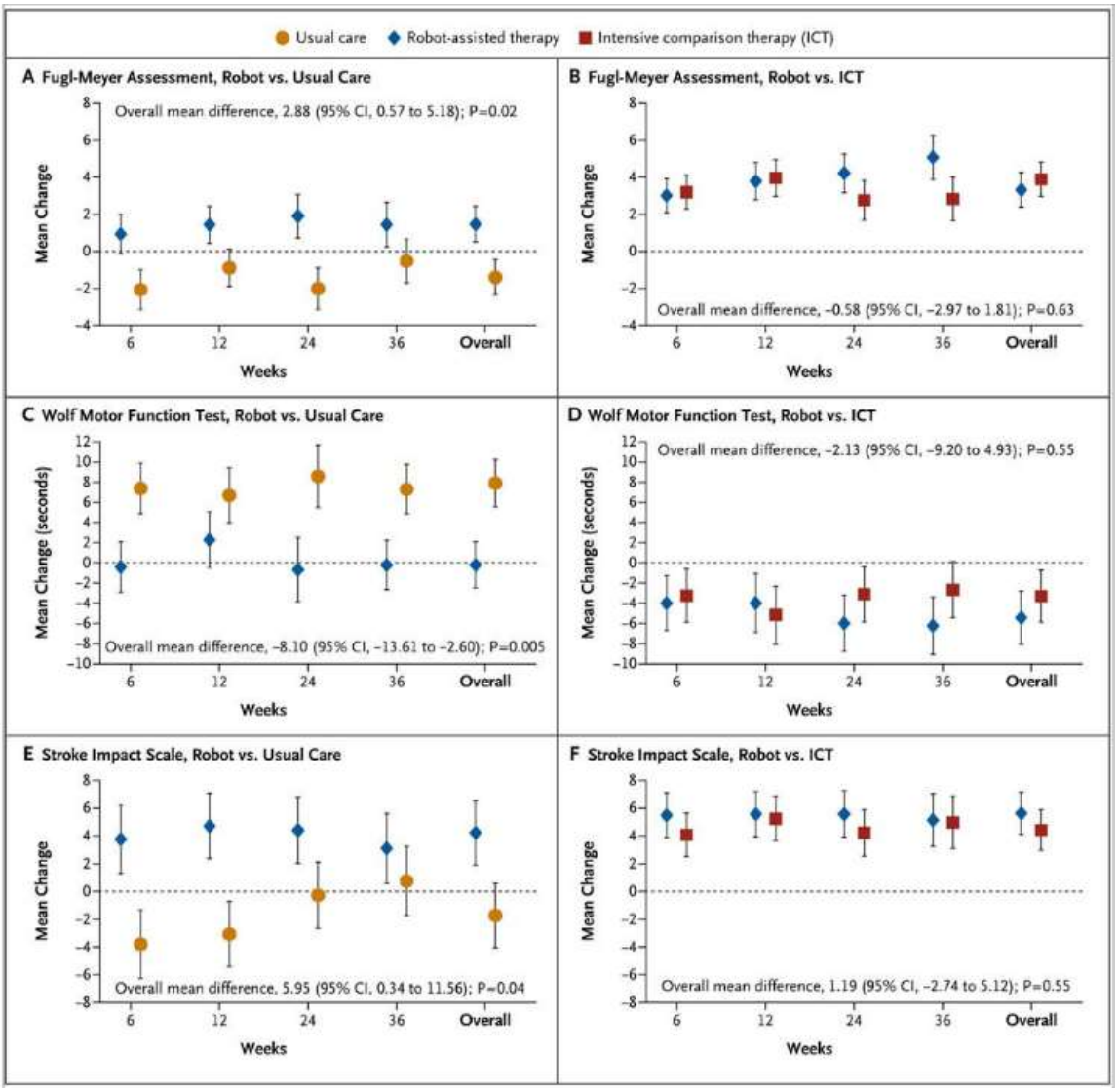

Figura 1 - Comparações entre os métodos de reabilitação robótica, reabilitação tradicional e reabilitação intensiva (LO et al., 2010)

2. OBJETIVOS

Este trabalho tem por objetivo final projetar um sistema de aquisição da posição de membros superiores de um paciente em tratamento. Para atingir este objetivo maior, alguns passos intermediários são propostos:

- Aquisição de dados através da visão fornecida pelo Kinect, utilizando o MATLAB;
- Aquisição da movimentação do objeto observado utilizando a IMU, através do Visual Studio;
- *Elaboração de uma máquina de estados responsável por decidir a fonte de posicionamento a ser utilizado;*
- *Criação de um software capaz de realizar os cálculos de deslocamento a partir dos dados recebidos dos dois dispositivos.*

3. Materiais e Métodos

Diversos dispositivos estão envolvidos na elaboração de um sistema preciso de acompanhamento dos movimentos dos pacientes. São necessários dispositivos de aquisição de dados, *softwares* e equipamentos capazes de realizar os cálculos de posicionamento e deslocamento.

Para a execução completa do trabalho, foram feitos softwares para aquisição dos dados fornecidos pelo Kinect e pela IMU. Para integração eficiente dessas duas fontes de dados, foi elaborado um *software* que integra as informações e toma as decisões de como as posições desejas serão calculadas.

Com o objetivo de melhorar o sistema de localização, foi criada uma máquina de estados capaz de decidir os momentos em que a visão fornecida pelo Kinect era falha. Para estes momentos é necessário que a IMU auxilie o programa a perceber qual foi o movimento executado pelo objeto observado.

3.1. Máquina de estados

Uma máquina de estados é um modelo matemático usado para descrever um sistema computacional. É uma máquina abstrata que contém um número finito de estados, sendo apenas um o atual. Em determinadas situações pode ocorrer uma mudança para outro estado, isto é chamado de transição. Quando a máquina começa a funcionar, ela pode estar em um dos seus estados iniciais. Outro conjunto de estados importantes são os estados finais. Se a máquina está em um dos estados finais quando a máquina para de funcionar, a entrada é aceita. A entrada é um conjunto de símbolos pré-determinados chamados de alfabeto. Se um símbolo específico em um determinado estado causa uma transação deste estado para outro estado, essa transação recebe o nome deste símbolo.

Usualmente, uma máquina de estados é representada por um diagrama como o mostrado na Figura 2. Neste exemplo é mostrado a máquina de estados para uma catraca de controle de acesso. A catraca permanece travada até que um cartão de acesso seja reconhecido, nesse momento, passa para o estado destravada e assim que um giro da catraca é feito, a máquina retorna para o estado travada. Os vértices representam estados, mostrados como círculos. As transições são os arcos com flechas apontando no sentido estado de origem para o estado de destino rotuladas com seus respectivos símbolos.

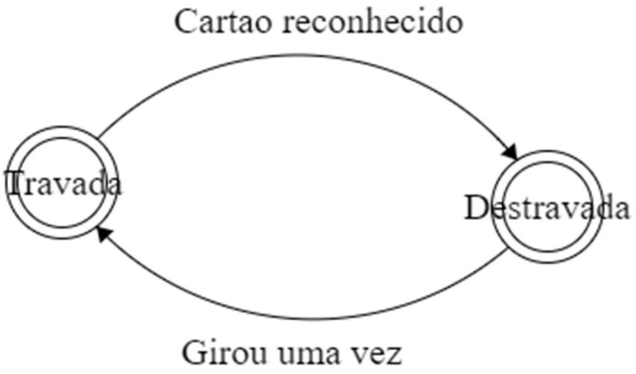

Figura 2 - Exemplo de Máquina de estados

Para a tomada de decisão, a utilização de uma máquina de estados como a mostrada na Figura 3 foi a abordagem escolhida para este trabalho por conta da simplicidade e do baixo custo computacional.

O estado q0 representa o estado inicial da máquina; q1 representa o estado em que a leitura do Kinect é suficiente para estabelecer a movimentação e os dados utilizados são os fornecidos por ele; q2 é o estado em que os dados da IMU são necessário, uma vez que o objeto observado não pode ser visto com clareza pelas câmeras do Kinect. Os estados q1 e 12 são estados finais aceitáveis e contém ações a serem tomadas. Já o estado q0 é apenas o estado inicial e serve apenas como ponto de partida para o programa.

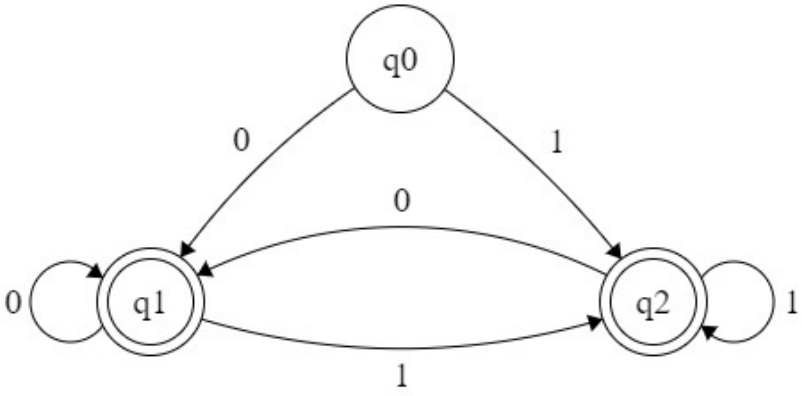

Figura 3 – Máquina de estados utilizada para a tomada de decisões

3.2. Manipulação de Dados

Para manipulação de dados são utilizados softwares matemáticos de e de desenvolvimento em diferentes linguagens de programação.

3.2.1. MATLAB

É uma linguagem de alto nível num ambiente interativo voltado à computação, visualização e programação numérica. Usando MATLAB, é possível analisar dados, desenvolver algoritmos e criar modelos e aplicações. A linguagem, ferramentas e funções matemáticas nativas permitem diferentes abordagens ao chegar mais rapidamente às soluções em comparação com planilhas ou linguagens de programação tradicionais, como C/C++ ou Java. Pode ser usado para uma série de aplicações, incluindo processamento de sinais e comunicação, processamento de vídeo e imagem, sistemas de controle, teste e medição, computação financeira e biologia computacional. É utilizado amplamente por engenheiros e cientistas na indústria e academicamente. É a linguagem de computação técnica (MATHWORKS, 2016a).

A linguagem usada no programa é chamada de M e, diferente de outras linguagens de programação, o MATLAB apresenta as soluções semelhantes à forma como são expressas matematicamente.

3.2.2. Visual Studio

É um pacote de programas de desenvolvimento da Microsoft. Utilizado principalmente para a linguagem C#, uma linguagem própria orientada a objetos.

O Visual Studio possui algumas versões no mercado, sendo a utilizada para este trabalho, a versão Visual Studio Community 2015. Versão de distribuição gratuita para fins individuais ou acadêmicos. Apresenta ferramentas para desenvolvimento de *software* para *desktop*, *web*, dispositivos moveis e computação em nuvem.

Possui diversas bibliotecas e componentes que facilitam o desenvolvimento e fazem com que as operações sejam simples, incluindo manipulação de dados e integração com outros programas como o próprio MATLAB.

3.3. Aquisição de Dados

3.3.1. IMU

Sensores que medem a posição inercial e velocidade em relação a um referencial pré-determinado são chamados de IMU (*Inertial Measerument Unit*) ou Unidade de Medida Inercial. Tal sensor possui, de maneira geral, giroscópios e acelerômetros internos que fornecem acelerações e movimento angular em relação aos eixos X, Y e Z. Para transformar os dados recebidos em dados da posição espacial X, Y e Z podem ser feitais integrais duplas em relação à aceleração ou, como foi feito neste trabalho, realizar o cálculo trigonométricos através dos ângulos *Yaw, Pitch* e *Row*. A Figura 4 mostra as variáveis angulares em relação a cada um dos eixos. Esse tipo de sensor é muito utilizado em aviões e navios, para que assim se mantenham informados da posição de navegação em que se encontram, por isso, historicamente, foram desenvolvidos para que funcionassem sem nenhum tipo de auxílio externo, ganhando então o nome de Unidade. A definição hoje perdeu um pouco da sua precisão e acaba aceitando também sensores com auxílio externo (DUDEK; JENKIN, 2008).

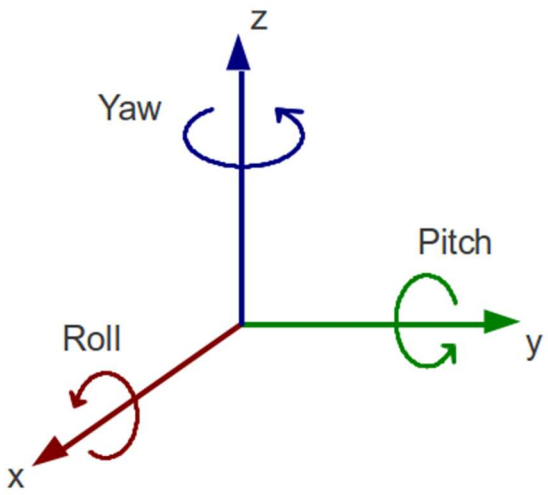

Figura 4 – Orientação de medições da IMU

A medida do vetor posição angular, é obtida através da integração das velocidades angulares, medidas através dos giroscópios embutidos no sensor. Da mesma maneira, a posição linear, é medida através da integração dupla das acelerações medidas pelos acelerômetros. Depois de um período de tempo suficiente, é necessário que toda IMU seja reiniciada e calibrada por um dispositivo externo, para que os erros sejam zerados (DUDEK; JENKIN, 2008).

É possível encontrar várias formas de IMUs no mercado, variando-se seu preço, precisão e qualidade das informações. A escolha deve ser feita de acordo com a aplicação em que se deseja utilizá-las. Em aeronaves, satélites, aplicações militares e

outras em que a precisão deve ser elevada, o preço da IMU também será alto. Em outros casos, em que a precisão pode ser menor, é possível encontrar opções com menor preço e qualidade dos dados, mas que se corretamente calibrados, atendem às necessidades da aplicação.

Usando o Visual Studio e a IMU Razor 9DoF (SPARKFUN, 2014) foi feito uma programa na linguagem C# capaz de receber os dados enviados pelo dispositivo conectado via USB. O que a IMU fornece são as informações de rotação ao redor do 3 eixos espaciais. As rotações são chamadas *yaw, pitch* e *row*, além das rotações, o dispositivos fornece as acelerações em cada um dos eixos além da direção e sentido dos campos magnéticos ao redor do dispositivo.

Inicialmente, o programa obtém as posições, em ângulos dos sensores de rotação, esses primeiros valores são utilizados como *offset,* dessa forma os valores obtidos a partir daí são a variação de rotação em cada eixo. O posicionamento da IMU em relação ao braço do usuário é mostrado na Figura 5. A distância entre o objeto a ser rastreado e o ombro do usuário é medido e chamado de L.

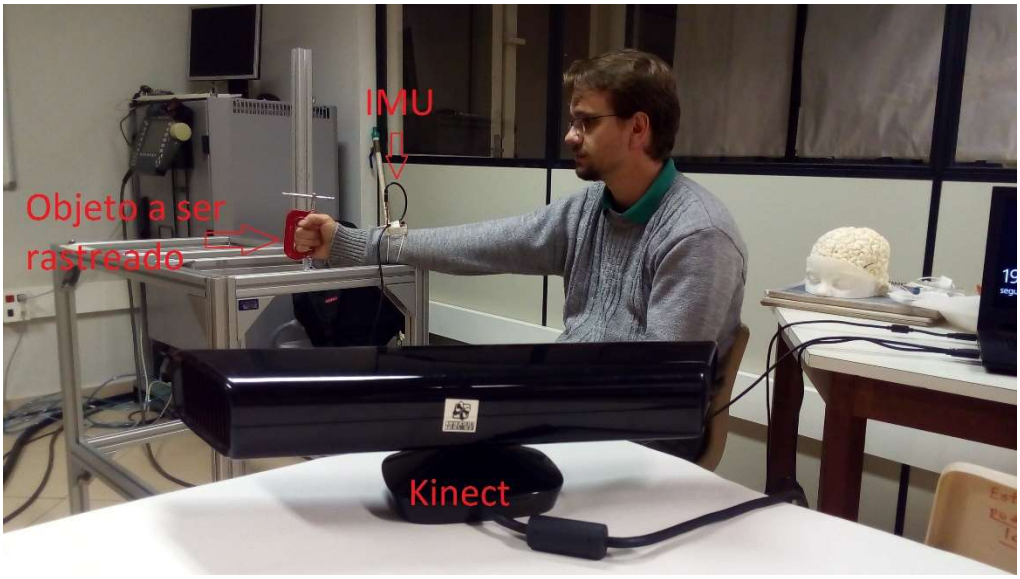

Figura 5 - Posicionamento da IMU no braço do usuário.

Através de cálculos geométricos, é possível calcular as posições X, Y e Z através dos ângulos de rotação *Yaw* (A)*, Pitch* (B) e *Row* (C)*,* obtidos da IMU. Os cálculos para cada eixo são:

$$X = L * Cos(A) * Cos(B) \quad (i)$$
$$Y = -L * Sen(A) * Cos(B) \quad (ii)$$
$$Z = -L * Sen(B) \quad (iii)$$

Sendo:

L: Tamanho do braço do paciente, passado como entrada do programa
A: Ângulo de rotação *yaw,* recebida da IMU
B: Ângulo de rotação *pitch,* recebida da IMU
C: Ângulo de rotação *row,* recebida da IMU

Com os cálculos feitos a cada movimentação, o programa utiliza a comunicação UDP para receber os dados de movimentação do Kinect recebido através do MATLAB. De acordo com a variável de decisão recebida (zero ou um), o programa decide de qual fonte virá a informação de localização. Caso a variável seja zero, o valor considerado é o do Kinect, caso seja 1, a localização é feita pela IMU.

Os valores recebidos de ambas as fontes foram salvas em arquivos distintos para posterior avaliação.

3.3.2. Kinect

É um dispositivo desenvolvido pela Microsoft juntamente com a Prime Sense - empresa israelense - para ser utilizado junto com o console Xbox-360, que permite que o jogador comande o videogame e os jogos utilizando apenas o movimentos corporais, sem uso de controles (MICROSOFT, 2016).

O dispositivo possui uma câmera RGB, para captura tradicional de imagens, um emissor e um sensor de infravermelho de profundidade em 3D para o mapeamento das distancias dos objetos à sua frente. Possui também microfones para aceitar comandos de voz do usuário.

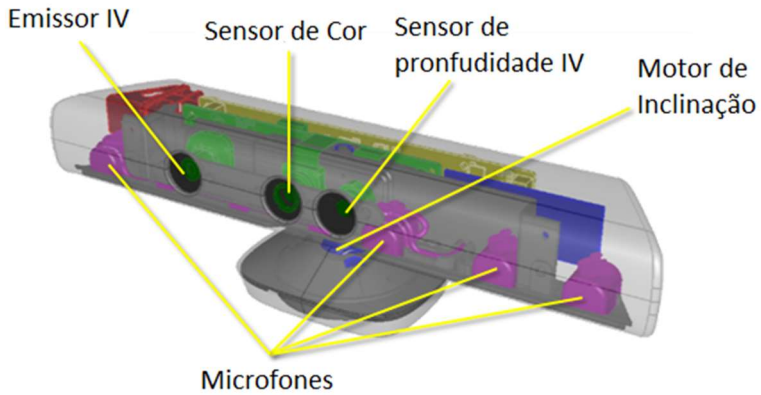

Figura 6 – Posicionamento de cada componente do Kinect (MICROSOFT, 2016)

O sensor de profundidade utiliza técnicas de visão computacional e processamento de imagem como profundidade por foco e comparação das imagens observadas de diferentes ângulos para obter uma nuvem de pontos com as suas respectivas coordenadas espaciais.

O Kinect tem um campo de visão de 43° na vertical e 57º na horizontal, isso limita uma área de captura, que fica mais restrita conforme aumenta o grau de precisão desejado. Quanto à distância, o Kinect consegue captar imagens entre 0.4m e 8m de distância, sendo considerada a distância efetiva entre 0.8m e 4m.

O campo de visão pode ser ajustado através do motor de inclinação que permite uma variação de ± 27°. Tanto a câmera RGB quanto a de profundidade capturam imagens a 30 quadros por segundo.

O MATLAB oferece um *add-on* para comunicação com o Kinect chamado *Microsoft Kinect for Windows Support from Image Acquisition Toolbox* que possibilita a aquisição de imagem do Kinect tanto da câmera RGB quanto da câmera de profundidade (MATHWORKS, 2016b).

Cada um dos quadros obtidos através da câmera RGB foi tratado para que o objeto desejado fosse detectado. A Figura 7 apresenta um exemplo de quadro antes do tratamento.

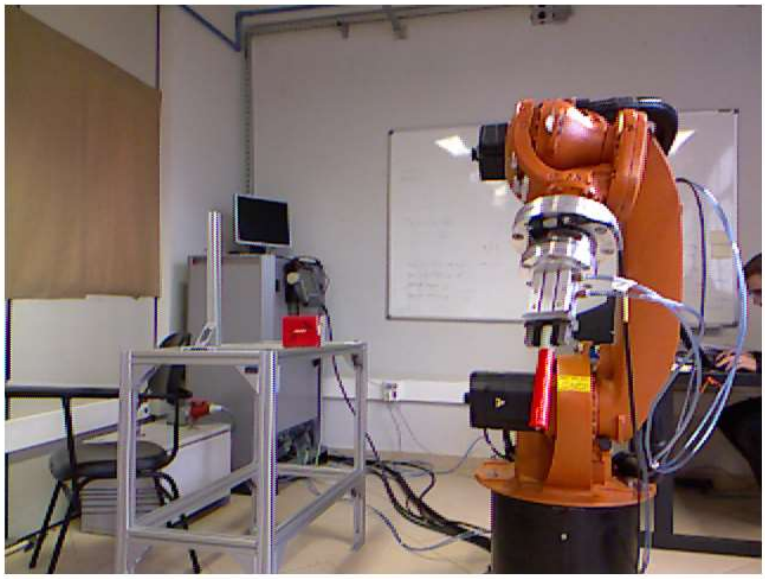

Figura 7 – Exemplo de quadro obtido pela câmera RGB.

O tratamento envolvido nessa imagem visa o rastreamento de objetos de cor vermelha. Foram aplicados filtros de cor e de média para diminuição de ruídos. Foram utilizadas funções do próprio MATLAB capazes de destacar os objetos de interesse com possível regulagem de brilho e intensidade de cor. Após aplicação dos filtros, foi possível obter a posição em *pixels* do objeto no plano da imagem.

Utilizando a câmera de profundidade, é possível obter uma matriz de distancias, em metros, para os eixos X, Y e Z. Dessa forma tem-se a posição espacial em relação ao Kinect, concluindo o rastreamento do objeto.

4. RESULTADOS E DISCUSSÕES

4.1. Tratamento da imagem

A partir da Figura 8 foram aplicados os filtros mencionados em 3.3.2.

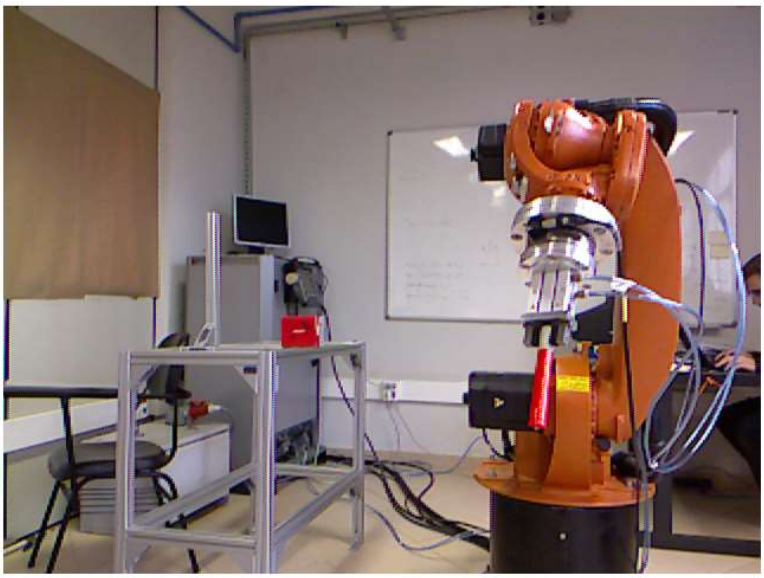

Figura 8 - Exemplo de quadro obtido pela câmera RGB.

O primeiro tratamento subtrai a imagem convertida para escalas de cinza da imagem com apenas a camada vermelha (cor escolhida para a detecção), resultando em uma imagem em que os únicos pontos brancos, são os objetos desejados. O resultado dessa primeira etapa é mostrado na Figura 9. A função responsável por isso é a "`diff_im = imsubtract(data(:,:,1), rgb2gray(data));`".

Figura 9 – Quadro após aplicação do primeiro filtro

A seguir é aplicado um filtro de mediana que faz com que cada *pixel* tenha o valor médio do pixels contidos em uma matriz 3x3 ao seu redor. Esse filtro faz com que a imagem tenha menos ruído do tipo “sal e pimenta” (quando *pixels* isolados apresentam valores muito mais altos ou mais mais baixo do que os *pixels* ao seu redor), principalmente nas bordas, deixando-as mais suaves. O resultado é mostrado na Figura 10. A chamada do filtro é feita através da função “`diff_im = medfilt2(diff_im, [3 3]);`”.

Figura 10 – Quadro após aplicação do segundo filtro

O próximo passo destaca o objeto desejado na imagem, transformando-a numa matriz binária. Esta operação é feita substituindo-se todos os pontos que apresentam um grau de brilho maior do que o passado por parâmetro para a função – objetos desejados, destacados pela primeira operação – por um na matriz da imagem e os pontos restantes por zero. Dessa forma, a matriz resultante apresenta apenas pontos com o maior grau de luminescência, um, nas posições do objetos desejados, enquanto todo o resto da imagem apresenta a menor luminescência possível, zero, ou seja, na cor preta. A Figura 11 mostra o resultado dessa etapa. Utilizando a função “`diff_im = im2bw(diff_im,0.18);`”.

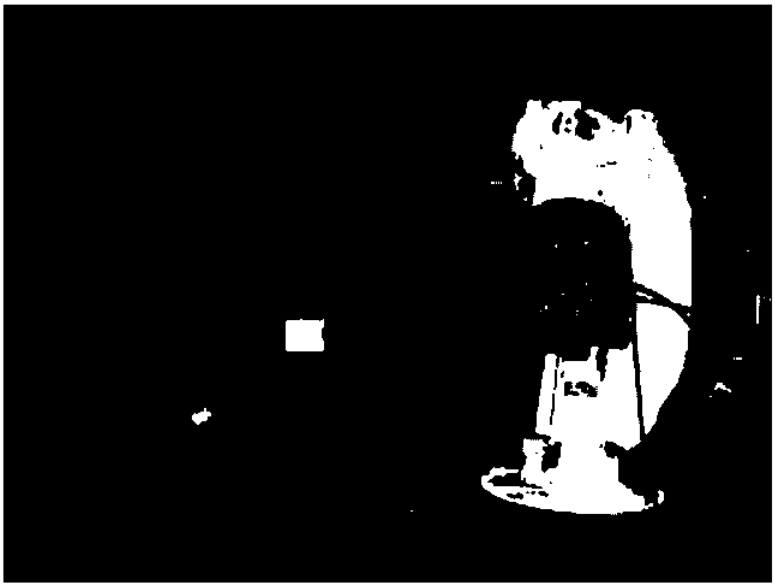

Figura 11 – Filtro de luminosidade aplicado.

Em seguida o tamanho mínimo do objeto a ser localizado na imagem é decidido através de uma função que elimina da matriz binária, objetos com menos pixel do que o decidido na chamada da função. Resultando em outra matriz binária, apenas com os pontos luminosos maiores do que o desejado. Através da função mostrada a seguir, é possível realizar a operação descrita. "`diff_im = bwareaopen(diff_im,200);`". Na Figura 12 é possível ver o resultado da operação.

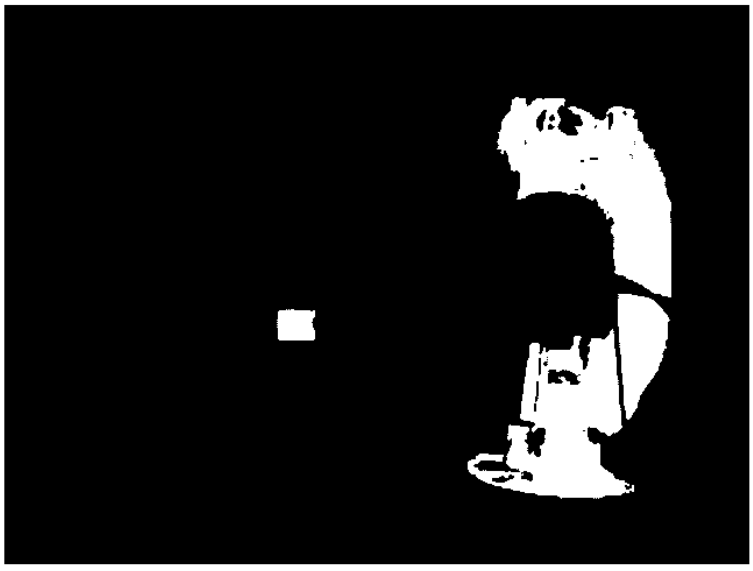

Figura 12 – Aplicação do último filtro e descarte de corpos menores do que o desejado.

Utilizando a função "`stats = regionprops(bw, 'Centroid');`" podemos obter o centro dos objetos destacados na imagem. Para que seja feito o rastreamento preciso, deve-se controlar o ambiente, garantindo-se que há apenas um objeto dentro do campo de visão do Kinect. Conseguindo assim, a posição, em pixels, do objeto desejado na imagem.

O mesmo quadro tratado foi observado através da câmera de profundidade. Dessa forma foi possível utilizar a função "`xyzPoints = depthToPointCloud(depthImage,depthDevice);`" que retorna uma matriz com dimensões 640X480x3 contendo as distancias em metros nos eixos X, Y e Z de cada pixel em relação ao Kinect. Essa matriz é chamada de matriz de profundidade. Quando não é possível obter a posição precisa de um pixel, o valor correspondente a sua posição é o símbolo NaN.

Após obter a posição espacial do objeto, uma conexão UDP local foi criada para comunicação com o programa em execução no Visual Studio que obtém as posições da IMU. O conteúdo da mensagem enviada é uma *string* contendo 4 informações: as posições X, Y e Z e uma variável definida em 0 quando a posição obtida pelo Kinect é satisfatória e 1 quando o valor de alguma das posições na matriz de posições for NaN.

4.2. Rastreamento do braço de um usuário

Em uma mesma situação de busca de dados, um usuário foi submetido a testes em frente ao Kinect e com uma IMU colocada em seu braço. A Figura 13 demonstra a situação de início do experimento.

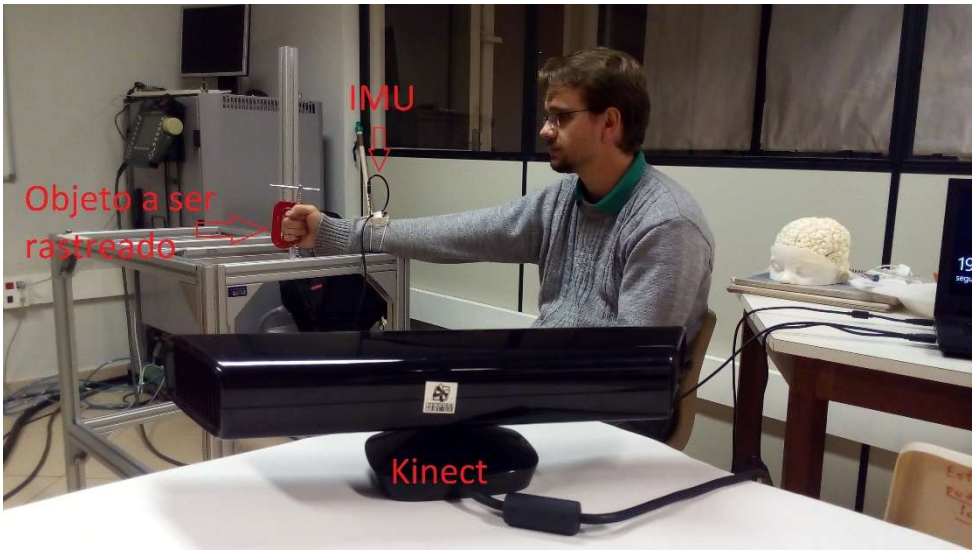

Figura 13 – Configuração inicial para realização do teste de localização com Kinect e IMU.

Para que a localização ocorra de maneira precisa, foi mantido na visão do Kinect apenas um objeto de cor vermelha. Esse objeto foi rastreado seguindo a rotina descrita em 4.1.

Após toda a rotina de tratamento o resultado final já com o objeto vermelho rastreado é mostrado na Figura 14.

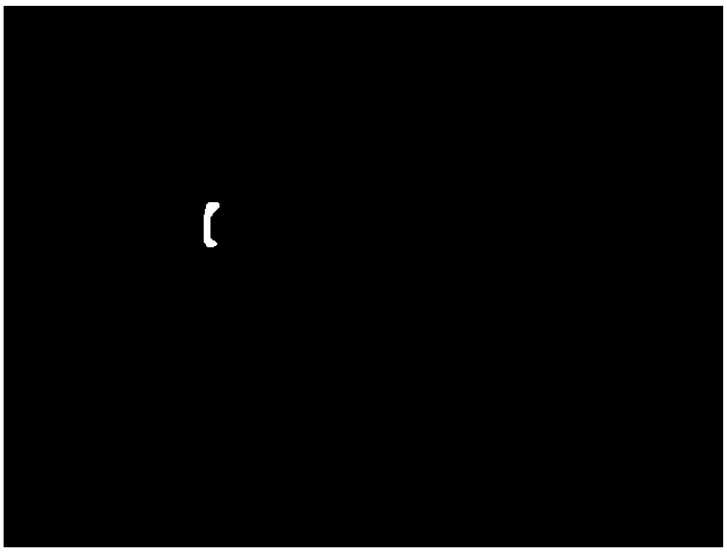

Figura 14 – Rastreamento do objeto na mão do paciente observado.

A aquisição de dados foi feita com movimentos nos 3 eixos. Para que fosse possível comparar os dois dispositivos, foi preciso igualar suas orientações. Sendo assim, as posições da IMU permanecem as mesmas e as do Kinect são alteradas para: X do Kinect = X da IMU; Y do Kinect = - Z da IMU; Z do Kinect = - Y da IMU. A origem do sistemas de coordenadas do Kinect é o primeiro ponto em que o objeto for rastreado, sendo todos os próximos pontos calculados em relação a ela. A origem do sistema de coordenadas da IMU, é o centro da mão do paciente, mesmo local em que o objeto vermelho se encontra, todos os pontos seguintes são calculados relativamente a ela. Desta forma, garante-se que os dois dispositivos utilizam mesma orientação e as origens estão no mesmo ponto no espaço. O comparativo eixo a eixo é mostrado na Figura 15, Figura 16 e Figura 17.

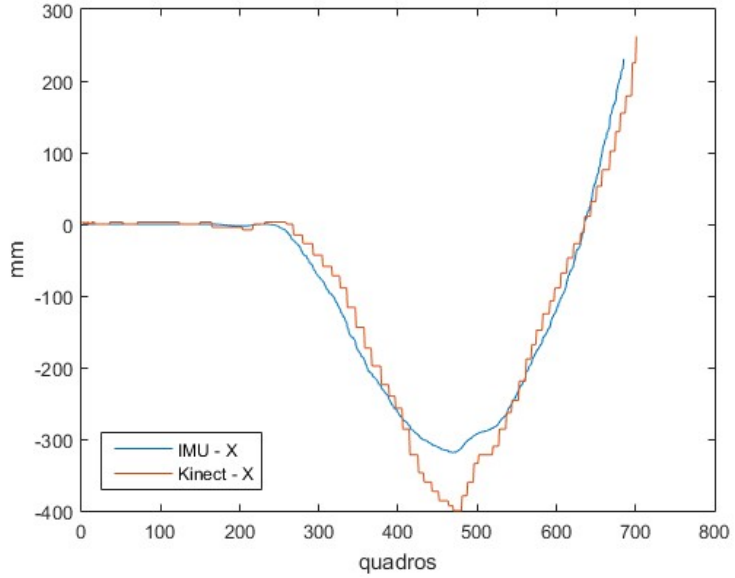

Figura 15 - Comparativo de dados obtidos pela IMU e pelo Kinect no eixo X

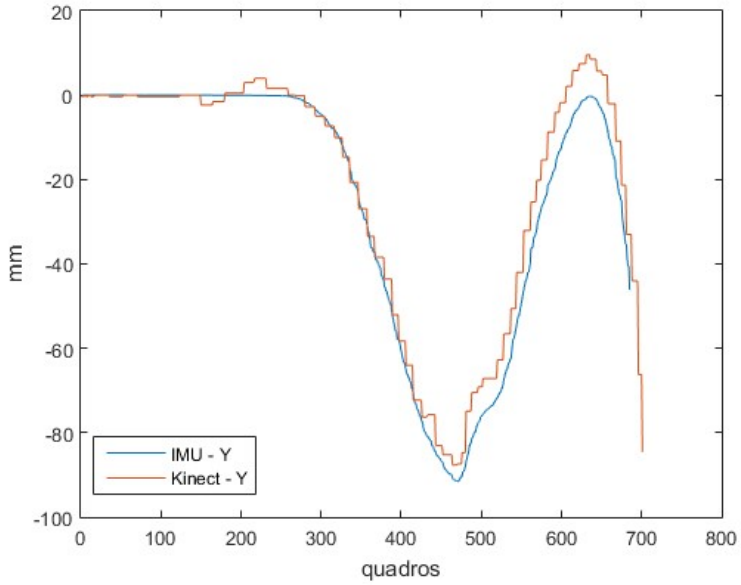

Figura 16 - Comparativo de dados obtidos pela IMU e pelo Kinect no eixo Y

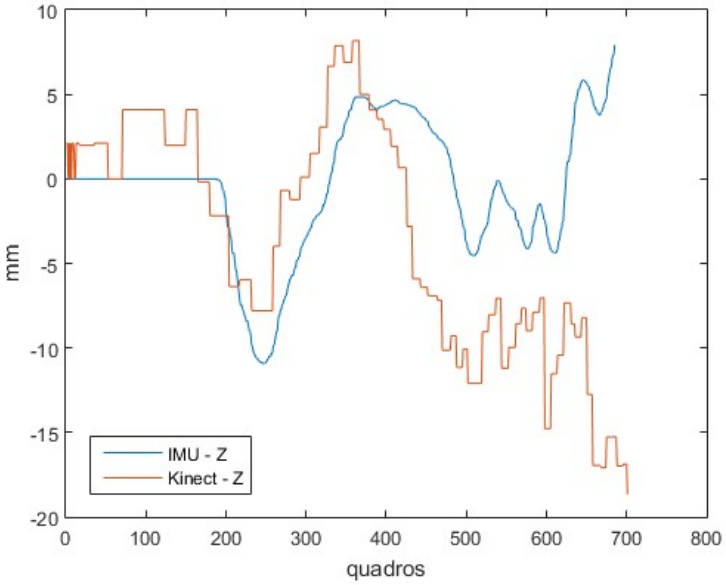

Figura 17 - Comparativo de dados obtidos pela IMU e pelo Kinect no eixo Z

O comparativo ponto a ponto mostra trechos em que a posição mostrada pelo Kinect sofreu oscilações e pôde ser considerado não satisfatório. Nesses pontos a IMU deve assumir e mostrar uma posição próxima a posição atual do objeto.

Para os 3 eixos, os 2 dispositivos estavam mostrando posições semelhantes, com diferenças que podem ser corrigidas por calibragem de acordo com a aplicação e o grau de precisão desejado.

É possível observar também, que a IMU fornece uma trajetória mais suave, enquanto o Kinect apresenta mudanças mais bruscas de direção, fato que causa os degraus nos gráficos. Os erros comparativos em cada eixo são mostrados na Figura 18, Figura 19 e Figura 20.

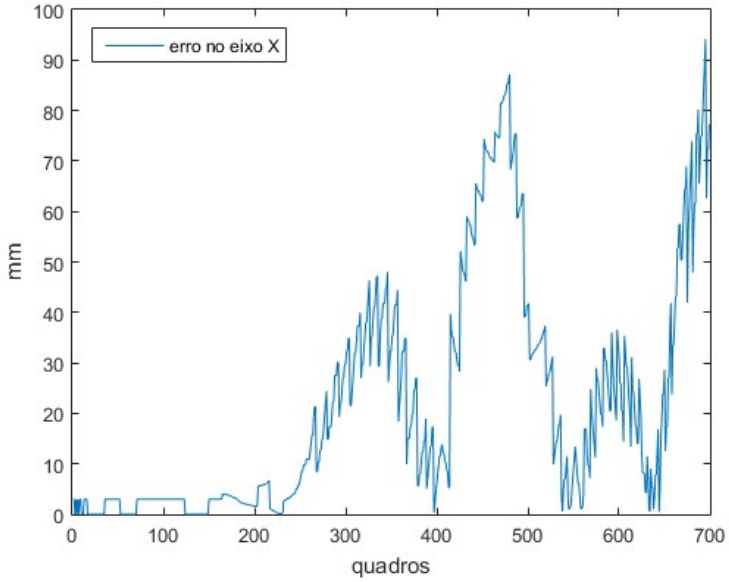

Figura 18 – Erro entre dispositivos no eixo X

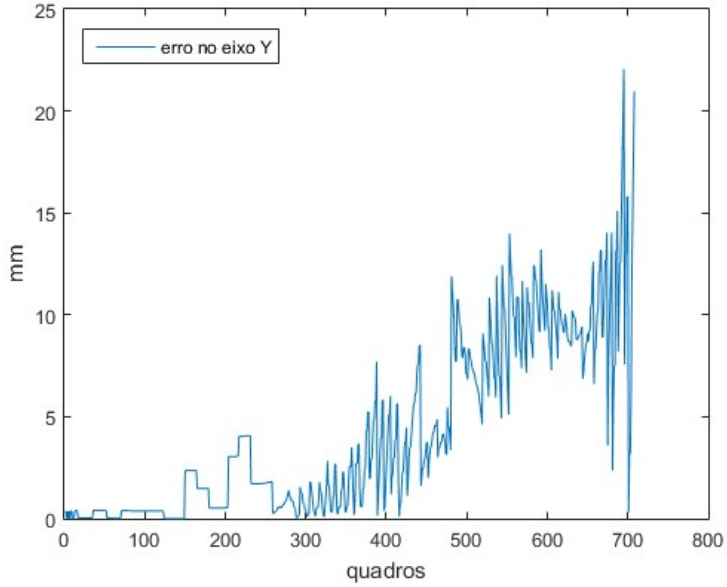

Figura 19 - Erro entre dispositivos no eixo Y

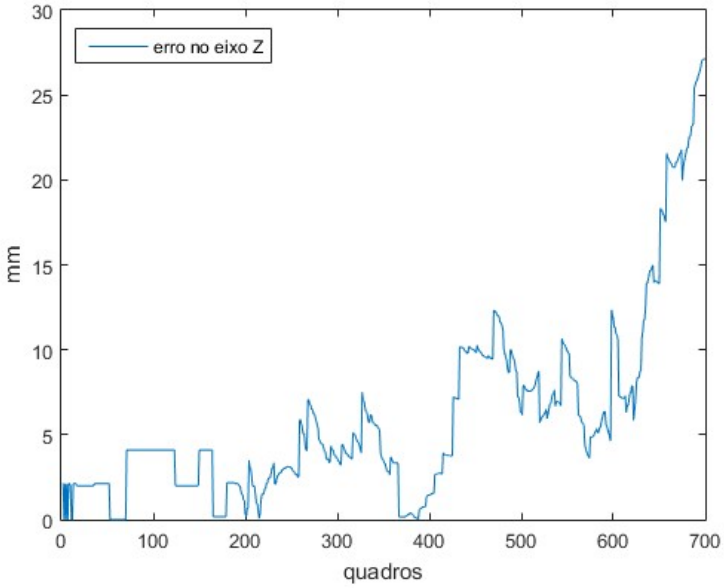

Figura 20 - Erro entre dispositivos no eixo Z.

5. CONCLUSÕES

Tarefas de reabilitação auxiliadas por robôs exigem uma maneira de monitorar a posição dos membros do paciente. Este trabalho propõe uma ferramenta para esse propósito, baseado em máquinas de estado. Utilizou-se um sensor de visão computacional e outro inercial, por se compreender que o primeiro pode falhar e o outro, assim, assume.

A utilização de algoritmos em Matlab e C# mostraram que a comunicação entre os dois não é problemática quando feita utilizando uma porta UDP para transmissão de dados em forma de *string*. As imagens obtidas mostram-se bastante promissoras, onde a confiabilidade nos sensores pode ser explorada. Deve-se notar a queda de desempenho do Kinect quando os objetos se distanciam.

Os erros nas direções ficaram menores que 100 mm em um ambiente com pouco controle de luz, o que pode então ser melhorado, mostrando a potencialidade do método. É possível utilizar mais IMU's para que a medição seja feita de forma mais precisa. Também é possível utilizar os dados da IMU para calcular o a velocidade e aceleração atual do objeto e fornecer uma posição calculada a partir do último ponto mostrado pelo Kinect.

6. REFERÊNCIAS BIBLIOGRÁFICAS

CAURIN, G. A. P. et al. Adaptive strategy for multi-user robotic rehabilitation games. **Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference**, v. 2011, p. 1395–8, jan. 2011.

DACIUK, J. **Finite State Automata**. Disponível em: <http://galaxy.eti.pg.gda.pl/katedry/kiw/pracownicy/Jan.Daciuk/personal/thesis/node12.html>. Acesso em: 25 maio. 2016.

DUDEK, G.; JENKIN, M. 20. Inertial Sensors, GPS, and Odometry. In: KHATIB, O.; SICILIANO, B. (Eds.). . **Springer Handbook of Robotics**. Würzburg: Springer, 2008. p. 477–490.

KREBS, H. I.; CAURIN, G. A. DE P.; BATTISTELLA, L. Rehabilitation robotics, orthotics, and prosthetics for the upper extremity. In: SELZER, M. et al. (Eds.). . **Textbook of Neural Repair and Rehabilitation**. 2nd. ed. Cambridge: Cambridge University Press, 2014. p. 760.

LO, A. C. et al. Robot-assisted therapy for long-term upper-limb impairment after stroke. **The New England journal of medicine**, v. 362, n. 19, p. 1772–83, 13 maio 2010.

MATHWORKS. **MATLAB Product Description**. Disponível em: <http://www.mathworks.com/help/matlab/learn_matlab/product-description.html>. Acesso em: 25 maio. 2016a.

MATHWORKS. **Microsoft Kinect for Windows Support from Image Acquisition Toolbox**. Disponível em: <http://www.mathworks.com/hardware-support/kinect-windows.html?requestedDomain=www.mathworks.com>. Acesso em: 25 maio. 2016b.

MICROSOFT. **Kinect for Windows Sensor Components and Specifications**. Disponível em: <https://msdn.microsoft.com/en-us/library/jj131033.aspx>. Acesso em: 25 maio. 2016.

SPARKFUN. **9 Degrees of Freedom - Razor IMU**. Disponível em: <www.sparkfun.com/products/10736>.