INTERFACE GESTUAL COM KINECT PARA SUBSTITUIÇÃO DE MOUSE

RAFAEL B. T. LIMA, ANDRÉ G. S. CONCEIÇÃO, PAULO C. M. A. FARIAS

Escola Politécnica, UniversidadeFederal da Bahia
R. Prof. Aristídes Novis, 2, Federação, 40210-630, Salvador, BA, Brasil
E-mails: rbritotl@hotmail.com, andre.gustavo@ufba.br, paulo.farias@ufba.br

Abstract— Interaction with computerized devices is recurrent in modern society. This interaction requires communication between parts involved; and gestural communication, inherent to humans, has great potential in this field of application. Electronic devices such as cellphones and tablets already use gesture commands in touchscreens, but there is plenty to do regarding tridimensional gesture. Some projects referenced in this article describe the application of sensors like Kinect in the translation of tridimensional gesture to specific software commands. To further this application and to demonstrate the power of gestural communication, an interface that allows the user to execute the main functions of a mouse through gesture was developed. It uses only the Kinect as sensor, giving the user freedom to control mouse-accessible programs without manipulating any instruments. The tools and the algorithm for cursor control are presented, and the interface performance is compared with other cursor manipulation tools.

Keywords— Gesture Interface, Human-Computer Interaction, Mouse Simulation, Kinect.

Resumo— A interação com aparelhos computadorizados é parte do cotidiano moderno. Essa interação requer comunicação entre as partes envolvidas, e a comunicação gestual, inerente ao ser humano, tem grande potencial de aplicações neste meio. Aparelhos eletrônicos como celulares e *tablets* já se utilizam de comandos gestuais realizados em telas sensíveis ao toque, mas há muito que se desenvolver em relação ao uso de gestos tridimensionais. Alguns projetos referenciados neste artigo demonstram a aplicabilidade de sensores como o Kinect na tradução de gestos tridimensionais em comandos para *softwares* específicos. Para ampliar essa aplicabilidade e demonstrar o poder do comando gestual, foi desenvolvida uma interface que permite executar no computador as principais funções de um *mouse* a partir de gestos do usuário. Utiliza-se apenas o Kinect como sensor, dando ao usuário a liberdade de controlar programas acessíveis por ponteiro sem a necessidade de manipular quaisquer instrumentos. São apresentadas as ferramentas utilizadas e o algoritmo para o controle do ponteiro, e o desempenho da interface é comparado com o de outras ferramentas de manipulação do ponteiro.

Palavras-chave— Interface gestual, Interação Humano-Computador, Simulação de *mouse*, Kinect.

1 Introdução

Computadores permeiam o cotidiano de grande parte da sociedade moderna. Mesmo indivíduos que não possuam computadores pessoais estão sujeitos à interação com dispositivos computadorizados (como terminais de autoatendimento em bancos e estacionamentos comerciais). A interação humano-computador requer comunicação entre as partes envolvidas, e esta comunicação pode tomar várias formas. Uma delas, que é inerente ao ser humano (mesmo em indivíduos cegos desde o nascimento) e frequentemente utilizada de modo espontâneo, é a comunicação gestual (Goldin-Meadow, 1999). O gesto é um recurso poderoso, capaz de expressar através de mínimos movimentos um desejo ou uma solicitação.

Motivado pelo potencial desse recurso, este artigo apresenta o desenvolvimento de uma interface gestual para substituição do *mouse* no uso de um computador, utilizando o sensor comercial Kinect para capturar informações tridimensionais dos movimentos do usuário. Telas sensíveis ao toque têm presença visivelmente crescente no mercado de *smartphones*, e têm o benefício de explicitar a relação entre os gestos do usuário e os efeitos obtidos no aparelho utilizado, enquanto a utilização do *mouse* não permite acompanhar o gesto e o movimento do ponteiro ao mesmo tempo (Wu *et al*., 2014). Entretanto, estas duas tecnologias se mantém atreladas à superfície, ignorando a natureza tridimensional dos gestos humanos. A implementação da interface gestual mencionada busca justamente demonstrar a aplicabilidade de ferramentas comerciais no avanço de uma interação humano-computador bidimensional para tridimensional.

Propostas de interfaces gestuais para interação tridimensional com computadores têm surgido com frequência. Jagodzinski e Wolski (2014) desenvolveram um laboratório químico virtual para estudantes de ensino médio e faculdade, com o uso do Kinect para permitir detecção e análise dos movimentos manuais dos usuários. Através de simulações realistas do manuseio de diversos itens do laboratório virtual, o sistema permite não só estudar o uso de produtos químicos sem riscos à saúde, como familiarizar o usuário com as devidas práticas em um laboratório real.

Ruppert *et al*. (2012) apresentam o desenvolvimento e uso de uma interface baseada em gestos para controlar o computador de uma sala operatória. Utilizando o Kinect para rastrear movimentos específicos da mão do médico, o sistema permite observar detalhes e navegar entre imagens de tomografia computadorizada tridimensional sem a necessidade de manusear o computador. Este é um benefício de grande valor em um ambiente que se deve manter esterilizado.

Kamel Boulos *et al*. (2011) e Roupé *et al*. (2013) descrevem o uso de sensores de profundidade como interface para o controle de simuladores de mapeamento tridimensional, com destaque para o controle do Google Earth utilizando o Kinect como sensor para gestos. São definidos gestos específicos para as funções desejáveis (e.g. mover apenas uma mão fechada para arrasto do mapa e aproximar ou afastar as mãos para controlar o *zoom*).

Nestes exemplos apresentados, bem como em diversos outros artigos atuais sobre interfaces humano-computador baseadas em gestos, o propósito é controlar *softwares* específicos (geralmente ferramentas de simulação tridimensional). Um passo fundamental que deve ser dado para a popularização deste tipo de interface é o tratamento mais genérico da ferramenta. Com esta visão, a interface desenvolvida e apresentada neste artigo contempla as funções básicas de um *mouse* (movimentação do ponteiro, cliques e arrasto), de modo que se possam utilizar quaisquer recursos acessíveis pelo ponteiro.

2 Ferramentas para Rastreamento

2.1 O sensor Kinect

O Kinect é um sensor de movimento que permite rastrear diversas juntas do corpo humano. Lançado em novembro de 2010 pela Microsoft como um acessório para o console de jogos Xbox, encontra aplicações nas mais diversas áreas tecnológicas que se utilizam da detecção de movimento, rastreamento de esqueleto ou reconhecimento facial. Dispondo de uma câmera sensível a cores, um sensor de profundidade (baseado em emissão e captação de sinais no espectro infravermelho), motor em sua base e uma matriz de microfones com cancelamento de ruído, o Kinect disponibiliza em sua interface USB diversos dados de rastreamento, incluindo coordenadas tridimensionais das juntas do corpo de um indivíduo (Figura 1) detectado pelo dispositivo (Microsoft, 2016).

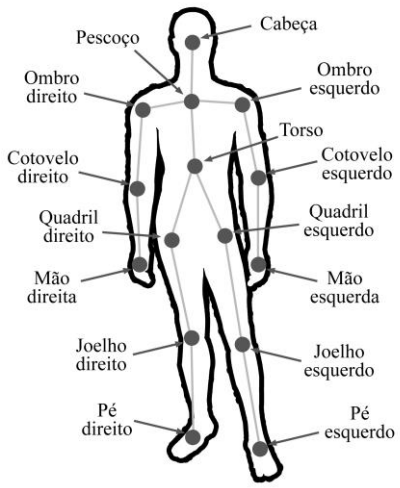

Figura 1. Indicação das juntas detectáveis pelo Kinect.

2.2 Acesso aos dados tridimensionais

O Kinect foi originalmente concebido para conexão ao Xbox, mas foram desenvolvidos *drivers* e *softwares* que permitem interagir com o sensor e capturar as informações de interesse. Duas ferramentas desenvolvidas nesse âmbito foram o OpenNI e o NiTE. OpenNI é um SDK (*Software Development Kit*, ou Conjunto para Desenvolvimento de *Software*) com funções para reconhecimento de gestos manuais, comandos de voz e rastreamento de movimentos corporais, enquanto o NiTE é um driver que agrega ao OpenNI o mapeamento de juntas corporais e outras funções para melhor identificação de gestos (Borenstein, 2012).

Instalando o OpenNI e o NiTE no ambiente do ROS (*Robot Operating System*, ou sistema operacional para robôs) de um computador com sistema operacional Ubuntu, pôde-se acessar informações sobre as juntas mapeadas pelo Kinect. ROS é uma estrutura de código aberto para desenvolvimento de *software* para sistemas robóticos, com uma vasta coleção de ferramentas e bibliotecas para aplicações de controle em plataformas computacionais (Quigley, Gerkey and Smart, 2015).

3 A Interface Gestual

Utilizando as ferramentas apresentadas na seção anterior, desenvolveu-se um algoritmo para captura em tempo real das coordenadas tridimensionais do torso, dos ombros e das mãos de um indivíduo posicionado em frente ao Kinect. A informação do torso foi utilizada para referenciar as demais coordenadas, enquanto que as posições dos ombros foram utilizadas para centralizar a área de mapeamento de gestos. Já as coordenadas das mãos foram diretamente relacionadas ao controle de posição (mão direita - MD) e dos botões (mão esquerda - ME) do *mouse*.

Para minimizar o ruído nos valores de coordenadas recebidas do Kinect, foi utilizada uma filtragem por média móvel. Segundo Smith (2003), este filtro é otimizado para a redução de ruído em sinais variantes no tempo, e pode ser definido conforme (1).

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j] \tag{1}$$

Nesta equação, x é o sinal de entrada e y é a saída filtrada. O filtro opera calculando a média de M pontos do sinal de entrada para obter cada ponto da saída. Para a interface desenvolvida, com uma média móvel de 15 pontos foi possível reduzir variações de até 8,9% no valor lido para menos de 2,2%.

3.1 Mapeamento das funções do mouse

Para a principal função do *mouse*, que é mover o ponteiro, foi associado o gesto de mover a mão direi-

ta. Assim, foi necessário definir uma área à frente do usuário na qual a posição da MD corresponderá de modo diretamente proporcional à posição do ponteiro na tela do computador. Durante a calibração da interface, o usuário deve posicionar a MD à sua frente, tal como ilustrado na Figura 2. As coordenadas da MD nesta pose serão associadas ao posicionamento do ponteiro no canto inferior direito da tela.

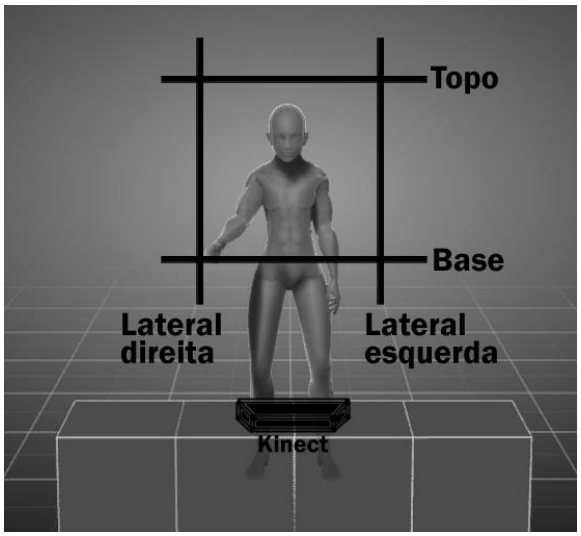

Figura 2. Delimitação da área de movimentação do ponteiro através de gestos.

As linhas em preto representam as fronteiras da área em que se pode mover o ponteiro, sendo o cruzamento entre "Lateral direita" e "Base" localizado na posição da MD durante o gesto de calibração. O centro da região é definido no ponto médio entre as coordenadas horizontais dos ombros do usuário, e as fronteiras "Topo" e "Lateral esquerda" representam respectivamente o limite superior da região e o limite à esquerda do usuário.

A Figura 3 ilustra uma visão superior de uma situação de uso da interface. Novamente, são destacadas por linhas pretas as fronteiras utilizadas para discernir qual gesto está sendo executado ao tratar as coordenadas fornecidas pelo Kinect. Neste caso, as fronteiras "Direita" e "Esquerda" têm o mesmo significado que "Lateral direita" e "Lateral esquerda" (respectivamente) na Figura 2. O usuário encontra-se diretamente em frente ao Kinect. Os gestos associados a cada função do *mouse* estão relacionados na Tabela 1, onde BE significa "botão esquerdo" e BD significa "botão direito".

O gesto de estender a mão esquerda em direção ao sensor é percebido quando a ME ultrapassa a fronteira "Base". Trazer a mão esquerda junto ao corpo significa, reciprocamente, mantê-la entre o torso e a fronteira "Base". Considera-se que a ME está estendida à esquerda do corpo quando ultrapassa a fronteira "Esquerda". Analogamente, o gesto de estender a ME para cima é identificado quando esta ultrapassa a fronteira "Topo".

Tabela 1. Gestos associados a cada função de interesse.

Função do *mouse*	**Gesto**
Mover ponteiro	Mover MD
Segurar BE	Estender ME em direção ao sensor
Soltar BE	Trazer ME junto ao corpo
Clique simples com BE	Trazer ME para lado direito do corpo
Clique duplo com BE	Estender ME à esquerda do corpo
Clique simples com BD	Estender ME para cima

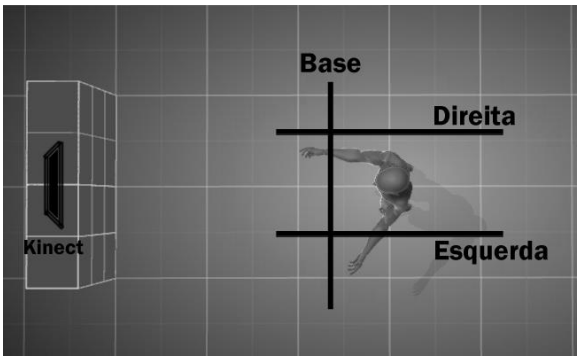

Figura 3. Fronteiras delimitadas para identificação de gestos – vista superior.

3.2 Controle do ponteiro

Uma vez identificado o gesto do usuário e determinada a respectiva função a executar, o sistema desenvolvido deve controlar o ponteiro na tela do computador para agir de acordo. Em ambientes Unix (como é o caso do Ubuntu), pode-se utilizar a ferramenta "xdotool" para simular a operação de teclado e *mouse* a partir de comandos simples (Gillmor, 2015). Inserindo estes comandos na programação da interface desenvolvida, pôde-se assumir o controle do ponteiro do computador. Os comandos utilizados e suas respectivas funções podem ser vistos na Tabela 2.

Tabela 2. Comandos utilizados da ferramenta xdotool.

Função do *mouse*	**Comando**
Mover ponteiro	xdotool mousemove x y
Segurar BE	xdotool mousedown 1
Soltar BE	xdotool mouseup 1
Clique simples com BE	xdotool click 1
Clique duplo com BE	xdotool click --repeat 2 1
Clique simples com BD	xdotool click 3

O algoritmo de operação da interface está representado no fluxograma da Figura 4.

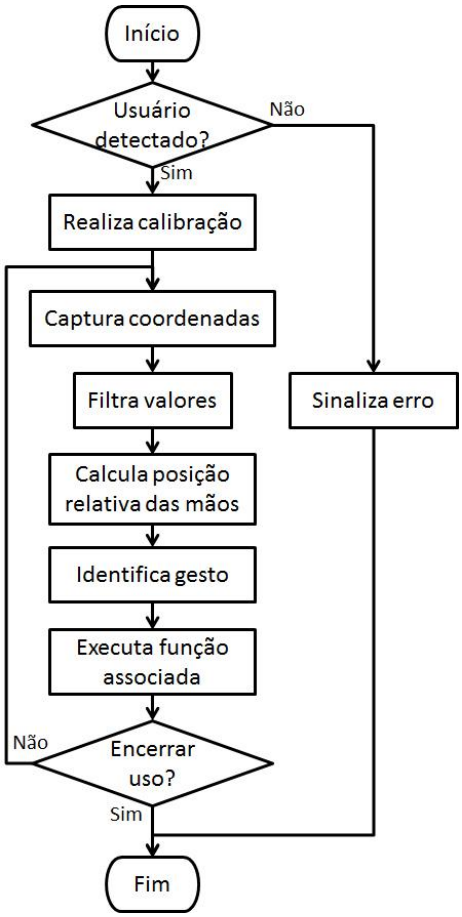

Figura 4. Fluxograma de operação da interface.

4 Validação Experimental e Resultados

Para a avaliação do desempenho da interface desenvolvida com o sensor Kinect, foram definidas três rotinas de teste: na primeira, em um programa de desenho, foi utilizado o ponteiro no modo pincel para traçar letras na tela; na segunda, foi medido o tempo médio de deslocamento do ponteiro desde o canto superior esquerdo até o centro da tela; e na terceira, foi avaliada a movimentação do ponteiro ao longo de um trajeto definido na tela. O primeiro teste foi realizado tanto com a interface desenvolvida como com um *mouse* comum. Os demais testes foram realizados com *mouse*, interface e com um apresentador multimídia (*pointer*) com controle tipo *joystick*.

O primeiro teste teve um propósito de avaliação qualitativa. Foram realizados dois tipos de traçado neste teste: à "mão livre", sem referências para auxiliar o desenho das letras, e "guiado", com as letras digitadas na tela para traçar por cima. Um dos resultados para o primeiro caso pode ser visto na Figura 5, e, para o segundo caso, na Figura 6. As duas figuras apresentam o traçado feito com *mouse* na parte superior e o feito com a interface na parte inferior.

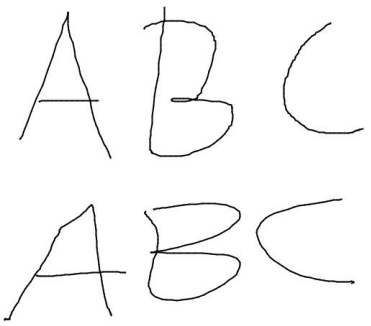

Figura 5. Traçado de letras sem referência.

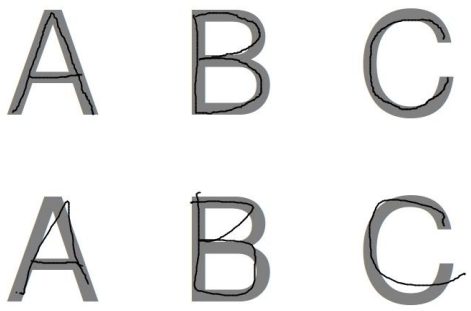

Figura 6. Traçado de letras com referência em cinza.

Para o segundo teste, foram realizadas 50 medições de tempo de deslocamento para cada um dos dispositivos (*mouse*, interface e *pointer*). Cada medida de tempo se iniciava ao atingir o canto superior esquerdo da tela, e se encerrava ao levar o ponteiro até um pequeno quadrado desenhado no centro da tela, como representado na Figura 7.

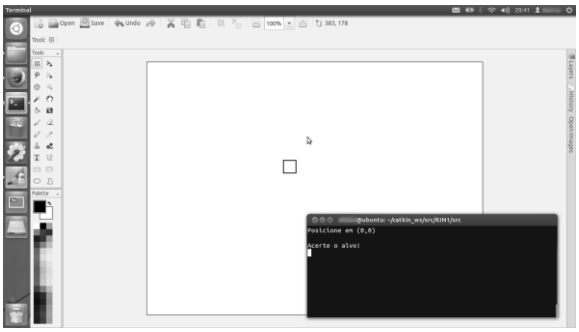

Figura 7. Tela para teste de tempo de deslocamento.

A média dos tempos obtidos por cada dispositivo pode ser vista na Tabela 3, juntamente com o maior e o menor tempo para cada um.

Tabela 3. Tempos de deslocamento para segundo teste.

	Tempo médio	**Maior tempo**	**Menor tempo**
Mouse	0,94s	1,45s	0,53s
Interface	3,19s	4,65s	1,83s
Pointer	3,55s	4,73s	2,17s

Para o terceiro teste, foi colocado um alvo próximo ao lado esquerdo da tela e duas linhas horizon-

tais atravessando a tela, delimitando um caminho a percorrer com o ponteiro até o lado direito. A tela apresentada durante esse teste pode ser vista na Figura 8. Foram realizadas 50 medidas de tempos de percurso desde o canto superior esquerdo, passando pelo alvo e seguindo pelo caminho delimitado até o lado direito da tela. Além disso, foram registrados os desvios que ocorram no caminho delimitado.

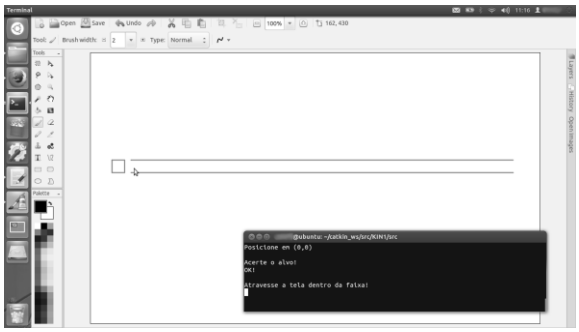

Figura 8. Tela para teste movimentação em trajeto definido.

Para este último teste, os resultados foram sintetizados na Tabela 4. O percentual de desvios foi calculado em relação à distância do caminho delimitado.

Tabela 4. Resultados para terceiro teste – valores médios.

	Tempo médio	**Maior tempo**	**Menor tempo**	**Percentual de desvios**
Mouse	3,47s	3,94s	2,96s	0,23%
Interface	6,81s	8,10s	5,85s	2,72%
Pointer	6,29s	7,19s	5,11s	1,56%

Tanto o segundo como o terceiro teste tiveram um propósito de avaliação quantitativa, coletando dados que permitam avaliar o desempenho da interface desenvolvida na manipulação do ponteiro. A comparação com o desempenho do *mouse* é esperada, visto que se propõe executar suas principais funções. Entretanto, resultados melhores são naturalmente esperados no uso do *mouse*, por ser uma ferramenta de uso difundido. Fez-se necessária a comparação com o *pointer* para obter dados em uma situação mais próxima à da interface desenvolvida: ambas as ferramentas permitem o uso à distância do computador e sem o uso de uma superfície de apoio.

5 Conclusão

Diante dos resultados do primeiro teste apresentado na seção 4, avalia-se que não houve qualquer perda de legibilidade quando comparado o traçado das letras feito através da interface com o feito utilizando *mouse*. Como se pode observar nas Figuras 5 e 6, o traçado com a interface foi efetuado com considerável qualidade, principalmente no segundo caso.

O segundo teste permitiu perceber que o uso do *mouse* ainda tem grande vantagem na agilidade de posicionamento do ponteiro, mas também deve ser ressaltado que a interface desenvolvida teve desempenho superior ao *pointer* com *joystick*, que é a ferramenta atualmente disponível no mercado para manipulação do ponteiro à distância. Já para o terceiro teste, o uso do *mouse* novamente teve resultados melhores, mas o *pointer* ultrapassou a interface no desempenho. Atribui-se essa mudança ao fato de o *joystick* facilitar a fixação do sentido de movimento desejado durante o percurso delimitado.

Ao permitir o controle do ponteiro e replicar as principais funções de um *mouse*, a interface desenvolvida diversifica as possibilidades de interação gestual com ferramentas computacionais. Com a tradução de gestos para comandos do próprio sistema operacional do computador, evita-se a limitação de quais programas se podem utilizar. Mesmo não atingindo o desempenho de um *mouse*, as comparações realizadas permitem avaliar que a ferramenta desenvolvida cumpre as funções esperadas, trazendo o diferencial de isentar o usuário da manipulação de controles ou sensores durante o uso da interface.

O trabalho realizado pode ter vários desdobramentos e permite diversas contribuições futuras, visando desde o aumento na precisão de posicionamento do ponteiro até a ampliação da gama de funções disponíveis. Podem ser avaliadas novas técnicas para interpretação dos dados fornecidos pelo Kinect, que não se restringem às coordenadas utilizadas atualmente; ou mesmo avaliar novos sensores disponíveis no mercado para substituir o Kinect e agregar outras funcionalidades.

Referências Bibliográficas

Borenstein, G. (2012). *Making things see*. Sebastopol: O'Reilly Media, 2012.

Gillmor, D. Ubuntu Manpage: xdotool - command-line X11 automation tool. Disponível em: <http://manpages.ubuntu.com/manpages/wily/man1/xdotool.1.html>. Acesso em: 2 abr. 2016.

Goldin-Meadow, S. Gesture in communication and thinking. Trends in Cognitive Sciences, Vol. 3, No. 11, 1999, pp. 419-429.

Jagodzinski, P., Wolski, R. (2014). Assessment of Application Technology of Natural User Interfaces in the Creation of a Virtual Chemical Laboratory. J Sci Educ Technol 2015 24:16–28

Kamel Boulos, M., Blanchard, B., Walker, C., Monteiro, J., Tripathy, A. and Gutierrez-Osuna, R. (2011). Web GIS in practice X: a Microsoft Kinect natural user interface for Google Earth navigation. International Journal of Health Geographics.

Microsoft. Kinect Effect. Disponível em: <http://www.xbox.com/pt-BR/Kinect/Kinect-Effect>. Acesso em: 3 abr. 2016.

Quigley, M., Gerkey, B. and Smart, W. (2015). *Programming robots with ROS*. O'Reilly Media, 2015.

Roupé, M., Bosch-Sijtsema, P., Johansson, M. (2013). Interactive navigation interface for Virtual Reality using the human body. Computers, Environment and Urban Systems 43, 2014, pp. 42–50.

Ruppert, G., Reis, L., Amorim, P., Moraes, T. and Silva, J. (2012). Touchless gesture user interface for interactive image visualization in urological surgery. World J Urol, 2012, 30:687–691

Smith, S. Digital signal processing: A Practical Guide for Engineers and Scientists. Amsterdam: Newnes, 2003.

Wu, Y., Yang, G. and Zhang, L. (2014). Mouse simulation in human–machine interface using kinect and 3 gear systems. International Journal of Modeling, Simulation, and Scientific Computing. Vol. 5, No. 4, 2014, 1450015 (13 pages).