Mapeamento de Projeções por Sistema Kinect-Projetor

Projection Mapping for a Kinect-Projector System

Thiago Motta*, Manuel Loaiza*, Luciano Soares‡, Alberto Raposo*

* Instituto Tecgraf de Desenvolvimento de Software Técnico-Científico
PUC-Rio – Pontifícia Universidade Católica do Rio de Janeiro
e-mail: {trmotta,manuel,abraposo}@tecgraf.puc-rio.br

‡ Escola de Engenharia
Insper Instituto de Ensino e Pesquisa
e-mail: lpsoares@insper.edu.br

***Resumo*— A realidade espacial aumentada permite a criação de ambientes virtuais projetados em superfícies irregulares. Isso, porém, demanda um conhecimento estensivo da área de calibração câmera-projetor. Esse artigo apresenta um framework desenvolvido para facilitar o uso dos dados obtidos pela calibração de um sistemas kinect-projetor em aplicações de visualização. Além disso, foram avaliadas diferentes técnicas de calibração para se obter as melhores abordagens.**

Palavras-chave—Calibração; Projeção Mapeada; Realidade Espacialmente Aumentada;

***Abstract*— Spatial augmented reality allows users to create a projected virtual environment on irregular surfaces. This demands an extensive knowledge in the Camera-Projector calibration area. This paper presents a framework developed to facilitate the use of data achieved from a calibration of a Kinect-Projector system in visualization applications. Additionally, different calibration techniques were evaluated in order to demonstrate the better approaches.**

Keywords—Calibration; Projection Mapping; Spatially Augmented Reality;

I. INTRODUÇÃO

Sistemas virtuais que aumentam as características encontradas em um mundo real cada vez mais demonstram sua relevância para diversas aplicações. Chamada de Realidade Aumentada [1], essa tecnologia combina conceitos de realidade virtual ao ambiente real, adicionando tradicionalmente gráficos e sons ao que vemos e escutamos para criar uma experiência única em interação virtual e simulação.

Projetos de rastreamento[2][3] foram desenvolvidos para permitir localizar geometricamente o usuário em um sistema de visualização. Usando um conjunto de câmeras, sensores com iluminação Infravermelha (IR) e marcadores para localizar pontos específicos no espaço, eles alteram o ambiente virtual baseado nos movimentos de uma pessoa. Esses sistemas retiram a necessidade do uso de periféricos menos intuitivos para interação, mas ainda dependem de monitores para apresentação dos dados.

Ambientes que só se preocupam com a visualização [4][5] usam um mapeamento estático da cena, medindo as dimensões de cada objeto deste ambiente para então reproduzi-lo no cenário virtual, processando os gráficos necessários sobre este e então projetando o resultado sobre o cenário real. Este ambiente precisaria de uma extensa calibração e não poderia ser reutilizado, possuindo também um custo elevado de produção.

Levando em consideração estes problemas, criou-se o conceito de mapeamento dinâmico/projeção mapeada[6]. Esse conceito define o resultado obtido ao calibrar um cenário sem ter de medi-lo, funcionando então para quaisquer objetos que estejam na cena, mesmo que estes sejam adicionados após a calibração.

Um framework foi desenvolvido, no contexto desta pesquisa, que aproveita os dados de calibração e gera um mapeamento dinâmico, em especial com o dispositivo de baixo custo Microsoft Kinect.

O framework desenvolvido pretende atender a usuários que desejam utilizar ou estudar uma ferramenta capaz de criar ambientes em Realidade Aumentada de forma dinâmica e simples. Um sistema de Projeção Mapeada pode ser usado em vários casos como:

- Apresentação de Produtos[7]
- Exposições[8].
- Propaganda[9].
- Performances ao vivo[10].
- Lazer[11][12].
- Medicina[13].
- Quaisquer indivíduos que desejem adicionar informações de forma dinâmica a um certo tipo de trabalho podem fazer uso desta tecnologia.

O framework proposto neste trabalho visa englobar as seguintes caracteristicas:

- Compreensão dos dados oriundos da calibração de um sistema Kinect-Projetor.
- Inserção de gráficos virtuais nas cenas capturadas.
- Reprojeção da cena aumentada

O objetivo final do framework é permitir que aplicações gráficas possam aproveitar os dados recuperados da calibração Kinect-projetor e aplicá-los à cena real, gerando uma visualização em Realidade Aumentada da cena capturada em tempo de execução.

O restante deste artigo está organizado da seguinte maneira. A seção II apresenta os trabalhos relacionados à Projeção Mapeada. Na seção III é descrito todo o processo necessário para se chegar à calibração de um sistema Kinect-

978-1-4799-4261-9/14 $31.00 © 2014 IEEE
DOI 10.1109/SVR.2014.53

Projetor. A seção IV detalha os estudos preliminares feitos para se atingir o resultado proposto. Na seção V são detalhadas as especificações do sistema. Por fim a seção VI apresenta o resultado final deste estudo.

II. TRABALHOS RELACIONADOS

A calibração do sistema Kinect-Projetor vem recebendo cada vez mais atenção, com projetos como:

- RGBDemo [14]
- ProCamCalib [15]
- Open-Light [16]
- CameraProjectorCalibration[17]

Os quatro projetos acima identificam a problemática citada anteriormente e visam encontrar os parâmetros intrínsecos e extrínsecos do projetor e das câmeras IR e RGB do Kinect.

A proposta inicial consistiu em pesquisar melhor os quatro métodos para que fosse possível entender as diferenças, os prós e os contras entre os processos de calibração usados.

Com a proposta inicial atingida, foi necessário criar uma solução capaz de cumprir o objetivo de mapeamento dinâmico, validando assim a calibração obtida.

A proposta de visualizações integradas com projeções não é algo completamente novo, porém, até então os objetos a serem projetados tinham de ser previamente conhecidos ou passar por uma fase de reconstrução que, além de comprometer a visualização em tempo real, também gerava artefatos durante o processo de escaneamento devido à técnica de luz estruturada visível.

Um dos trabalhos mais tradicionais em projetar imagens sobre superfícies irregulares é o iLamps [18], onde são apresentadas técnicas para projeções adaptativas em superfícies não planares usando algoritmos de projeção com texturas. Já o trabalho desenvolvido no Sunnybrook Health Sciences Center, usa dispositivos de rastreamento sem marcadores para poder ver e navegar em dados de tomografia e ressonância magnética sem o uso de teclados e mouses [19], usando gestos com as mãos para navegar nas camadas de dados a serem visualizados.

A projeção de imagens sobre o corpo de uma pessoa com o apoio do Kinect é apresentado em um trabalho de uma massagem futurista [20]. Nela, o massagista projeta padrões de fluxos de linhas sobre o corpo da pessoa, contudo não existe um casamento preciso da imagem, tão pouco as imagens projetadas são para fins de alguma análise.

Apesar da variedade de trabalhos existentes na área de visualizações integradas com projeções, foram encontrados poucos sistemas capazes de solucionar a problemática da projeção mapeada dinâmica usando o Kinect e que fornecessem a flexibilidade necessária para funcionar com os diferentes métodos de calibração encontrados.

III. CALIBRAÇÕES DOS SISTEMAS DE COORDENADAS

Antes de se realizar um mapeamento, é necessário calibrar um sistema, sendo este usualmente constituído por um par câmera-projetor.

A calibração de câmeras constitui uma das maiores dificuldades da Visão Computacional, e tem por finalidade extrair os parâmetros extrínsecos e intrínsecos de uma câmera. Os parâmetros obtidos definem a posição 3D, rotação, distância focal, centro de imagem e coeficientes de distorção que a câmera possui. Com esses dados, torna-se possível a inserção de modelos virtuais em um ambiente real capturado pela câmera.

A. Calibração de uma câmera

O método mais básico é o da calibração de uma câmera, onde é necessário estimar na imagem obtida pela câmera uma série de pontos p1, p2, ..., pn que correspondam a pontos conhecidos P1, P2, ..., Pn do espaço tridimensional. Com os pontos correspondidos, calculam-se os parâmetros intrínsecos e extrínsecos tais que os pontos virtuais p'1, p'2, ..., p'n se sobreponham da melhor forma aos pontos observados. Para representar os pontos conhecidos geralmente se usa um padrão de tabuleiro de xadrez rígido com dimensões conhecidas. Esse padrão é movimentado em frente à câmera enquanto diversas snapshots são tiradas, de forma a guardar diferentes posições e rotações do padrão. Os algoritmos mais usados para se fazer este processamento são os desenvolvidos por Zhang [21] e Tsai [22].

A calibração é realizada usando como base os seguintes sistemas de coordenadas, de forma que a transformação da câmera possa ser definida como o conjunto de transformações realizadas entre estes sistemas:

Sistema de Coordenadas do Mundo (SCM)
Sistema de Coordenadas da Câmera (SCC)
Sistema de Coordenadas de Imagem (SCI)
Sistema de Coordenadas em Pixel (SCP)

$$[p] \simeq \underbrace{\begin{bmatrix} s_x & \tau & u_c \\ 0 & s_y & v_c \\ 0 & 0 & 1 \end{bmatrix}}_{(A)} \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{(B)} \underbrace{\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{(C)} [P];$$

Equação 1 – TransformaçãoAfim

A Equação 1 define as seguintes transformações:

(C) A multiplicação pela matriz RT define a transformação de SCM para SCC, chamada de Mudança de Referencial 3D.

(B) Esta multiplicação define a transformação de SCC para SCI, chamada de Projeção Perspectiva.

(A) A multiplicação da terceira matriz define a transformação de SCI para SCP, chamada de Transformação Afim, onde:

f é a distância focal;

s_x e s_y são o número de pixels por unidade de comprimento em ambas direções;

u_c e v_c são as coordenadas de projeção ortogonal do centro óptico no plano de projeção;

τ é dada pela tangente do ângulo que as colunas da matriz de pixels formam com a perpendicular com as linhas;

R e T, correspondem a matriz de rotação e o vetor de translação.

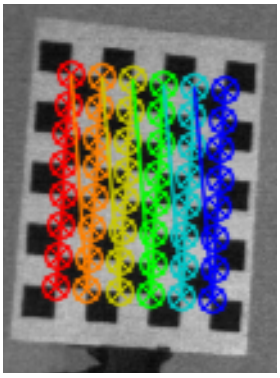

Figura 1 - Calibração sobre um padrão de xadrez

O resultado final é o representado pela Figura 1, onde é possível observar que os pontos virtuais (os círculos e linhas coloridos) se sobrepõem aos cantos internos do padrão. Nota-se que mesmo que o padrão seja movido ou rotacionado, os pontos virtuais continuarão a se sobrepor à este.

Apesar de funcionar para uma configuração simples, tal abordagem não é suficiente para atingir o objetivo proposto neste trabalho, porém ela serve como base para que se possa atingir um nível mais alto de complexidade.

Um sistema de mapeamento dinâmico consiste primeiramente de uma câmera, usada para que se possa obter a correspondência entre os pontos no SCM e SCP, porém ainda é necessário apresentar dados virtuais de volta para o usuário e sobre a cena calibrada. Para isso é necessário usar um par câmera-projetor, em que um projetor pode ser interpretado como o dual de uma câmera, necessitando assim da calibração de duas câmeras.

B. Calibração de duas câmeras

Notando que cada câmera possui seus próprios parâmetros de calibração,definimos como SCC1 e SCC2 o sistema de coordenadas de cada câmera, denotados por $[\,R_1 \quad T_1\,]$, e por $[\,R_2 \quad T_2\,]$, respectivamente, em relação a um mesmo SCM utilizado na calibração.

A transformação dos parâmetros extrínsecos da primeira câmera para si própria é dada pela multiplicaçao por $[\,I \quad 0\,]$, já que o sistema se encontra na posição desejada.Tendo que a calibração de uma câmera é feita encontrando os pontos do SCC em relação ao SCM, analogamente, a calibração de duas câmeras é feita encontrando-se os pontos do SCC2 em relação ao SCC1.

A obtenção dos parâmetros extrínsecos da segunda câmera em relação à primeira é feito ao encontrar no SCC2 um ponto conhecido de SCC1, que se resume a encontrar a matriz de rotação e translação de um sistema em relação a outro. Isto é feito ao gerar a mudança de SCC1 para SCM para então mudar de SCM para SCC2, onde a primeira mudança é feita pela multiplicação por $\begin{bmatrix} R_1^t & -R_1^t T_1 \\ 0 & 1 \end{bmatrix}$ e a segunda pela multiplicação de $\begin{bmatrix} R_2 & T_2 \\ 0 & 1 \end{bmatrix}$. Ou seja, as coordenadas P1 e P2 de um ponto do espaço com relação ao SCC1 e SCC2 satisfazem as relações $P2 = RP1 + T$ e $P1 = R^t P2 - R^t T$.

C. Calibração de um sistema Câmera-Projetor

A óptica interna de uma câmera e de um projetor são semelhantes e ambos podem ser modelados da mesma maneira.

A calibração câmera-projetor é feita de forma semelhante a uma calibração entre duas câmeras, onde ambas as câmeras ou o par câmera-projetor devem ser calibrados em relação a um mesmo sistema de referência do mundo. Com a calibração é possível correlacionar os pontos do mundo com os pontos capturados, ou seja, ambas as câmeras são posicionadas de forma a capturar a mesma cena.

A diferença entre calibrar duas câmeras e calibrar um par projetor-câmera é que o projetor não captura pontos em um SCM, assim a ideia é projetar uma imagem conhecida e determinar os pontos encontrados na cena, conforme é mostrado na Figura 2.

Uma forma conveniente de realizar isso é usar um padrão de xadrez virtual (projetado) sobre o mesmo plano em que se encontra um padrão de xadrez físico (impresso). Para obter as coordenadas no SCM de um ponto visto pelo projetor se aplica a transformação inversa do projetor à transformação da câmera.

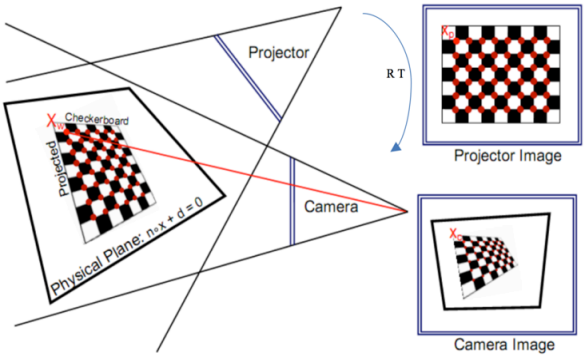

Figura 2 - Representação de um Sistema Câmera-Projetor[16]

A calibração do sistema projetor-câmera permite a identificação de pontos 3D em uma cena e a reprojeção destes sobre eles mesmos. Entretanto, para se reproduzir fielmente uma imagem em 2D sobre um objeto real em 3D, é necessário que se compute a malha extraída da cena no mundo real, para então realizar a aplicaçãoda imagem em 2D sobre esta malha e só então projetar este resultado. Para tornar o caminho mais curto, foi escolhido como câmera o dispositivo Kinect que já possui um sistema de captura de profundidade da imagem.

D. Dispositivo de captura RGB-D - Kinect

O Kinect é um sensor de movimento desenvolvido primeiramente como um dispositivo de interação natural para jogos. Dado seu baixo custo em comparação aos sistemas comerciais RGB-D, onde D significa distância (Depth), este sistema vem sendo usado para fins diversos, incluindo mapeamento e modelagem 3D[23].

O Kinect possui três elementos chave que possibilitam a captura de uma imagem de profundidade e de cor simultaneamente: Câmera RGB de 640 x 480, Câmera IR de 640 x 480, e um Projetor IR.

Com o par câmera-projetor IR e um sistema chamado Light Coding [24], o Kinect é capaz de detectar a distância entre um ponto e ele mesmo, tornando-se assim um dispositivo de medição 3D em tempo real. O Light Coding codifica informações em padrões de luz na saída do projetor IR, o que, ao ser projetado sobre uma superfície qualquer, gera uma deformação do padrão projetado, gerando assim informações necessárias para o cálculo da distância.

E. Calibração de um sistema Kinect-Projetor

O uso do dispositivo Kinect traz uma desvantagem: ele aumenta o número de câmeras do sistema. O que antes se resumia em:

1) Capturar dados da cena fazendo uso de uma câmera RGB, resultando em dados em 2D

2) Reprojetar os dados usando um projetor.

Agora se traduz em:

1) Capturar os dados da cena com uma câmera IR, resultando assim em pontos 3D.

2) Transformá-los para o espaço da câmera RGB

3) Aplicar uma nova transformação para enfim chegar ao espaço do projetor.

4) Reprojetar os dados usando um projetor.

O Kinect possui uma calibração padrão de seus sensores, entretanto esta calibração fornece somente uma aproximação da correlação dos pontos de diferentes sensores e por este motivo é necessário realizar uma calibração entre eles.

Apesar de seguir a mesma lógica da calibração entre duas câmeras já apresentada, pelo fato de uma das câmeras capturar somente IR, uma preocupação extra é adicionada: é necessário separar as imagens obtidas para a calibração em dois grupos diferentes, o grupo que está com o emissor IR obstruído, ou seja, que não possui iluminação IR, e o grupo que possui iluminação IR.

IV. ESTUDOS PRELIMINARES, CONCEITUAIS E DE TECNOLOGIA

Para fazer o uso correto do Kinect, as seguintes bibliotecas foram pesquisadas: OpenNI [25], Microsoft Kinect SDK [26] e Open Kinect [27].

O Microsoft Kinect SDK é a biblioteca mais modular para se usar no desenvolvimento de novas aplicações. Com módulos muito bem definidos, uma gama de exemplos disponíveis e documentação completa. Entretanto, ela é muito mais complexa na linguagem C++, linguagem usada em todo o desenvolvimento do programa para esta pesquisa, do que em C#, além disso, foram encontrados poucos exemplos relacionados ao escopo deste trabalho que usavam esta biblioteca.

O Open Kinect, apesar de ter sido uma das primeiras bibliotecas a oferecer suporte ao Kinect, não possui uma comunidade muito grande, com cerca de 2000 usuários ativos, e seu avanço depende somente dos poucos usuários que queiram aprimorá-la.

Por fim, temos a biblioteca do OpenNI, criada e desenvolvida pelos criadores por trás da tecnologia do Kinect, a PrimeSense [24]. Foi a biblioteca mais fácil de ser aprendida, além de possuir diversos programas de demonstração e um manual com detalhes e códigos para se entender como usar os principais recursos do Kinect. Por todos os motivos acima citados, essa foi a biblioteca escolhida para o desenvolvimento desta pesquisa.

Uma vez com o SDK de uso do Kinect escolhido, foi necessário pesquisar sobre os projetos existentes que solucionavam a calibração do sistema Kinect-Projetor.

RGBDemo

O RGBDemo [14] fornece, entre outras funcionalidades, a de calibração das câmeras do Kinect e a calibração de um sistema câmera-projetor. O driver usado para a calibração do Kinect foi o libfreenect, ao passo que o driver usado para a calibração do sistema Kinect-Projetor foi o OpenNI. Era necessário usar ambos os drivers, uma vez que o OpenNI não fornece opção para visualização simultânea entre as câmeras de RGB e o dado da câmera IR. O arquivo resultante da calibração é do formato YAML[28] e possui todos os parâmetros intrínsecos e extrínsecos obtidos. O tempo levado para a calibração varia entre 5 e 15 minutos para um conjunto de 30 imagens, que nem sempre continha um resultado coerente, sendo assim necessário repetir os passos anteriores diversas vezes. Ao final de todo o processo, o tempo consumido podia variar de 40 minutos a 1 hora e meia, sendo que o tempo levado pelo algoritmo para processar todas as imagens e gerar as matrizes de calibração durava cerca de 10 minutos enquanto o tempo restante foi usado para o posicionamento do tabuleiro de xadrez.

CameraProjectorCalibration

A proposta deste método [17] é de ser totalmente automatizado, não precisando assim que o usuário entre com comandos pelo teclado para selecionar uma imagem, consumindo muito menos tempo que o RGBDemo. O driver usado para este método foi o Microsoft Kinect SDK, uma vez que o programa funciona em conjunto com o OpenFrameworks [29] e não foi possível retirar tal dependência, fazendo assim com que fosse necessário emular [30] a câmera do Kinect como uma webcam para que o CameraProjectorCalibration o interpretasse como uma fonte de vídeo e assim usar as informações da câmera RGB como input para a calibração do sistema Kinect-Projetor. Apesar deste método possuir uma calibração entre um par de câmeras, não foi possível utilizá-la, uma vez que a integração com um driver do Kinect através do serviço de webcam não fornecia o resultado da câmera IR, portanto, a calibração de câmera usada aqui foi obtida com o RGBDemo.

O tempo total de calibração varia de 25 minutos a 50 minutos, mas este método atingiu um resultado coerente já na primeira ou segunda tentativa, garantindo uma variação de profundidade de quase dois metros. Apesar do tempo total

de calibração ser mais baixo em relação ao RGBDemo, o tempo de processamento do algoritmo deste método é superior, sendo responsável por ao menos 15 minutos do tempo total.

Apesar das diferenças enter o RGBDemo e o CameraProjectorCalibration, as seguintes conclusões foram observadas sobre ambas as soluções:

Não é recomendado realizar a calibração com mais de 30 imagens, uma vez que a complexidade das equações usadas para encontrar os parâmetros intrínsecos e extrínsecos aumenta conforme o número de imagens obtidas, precisando assim de um maior tempo de calibração e possuindo maior chance de gerar um resultado ruim.

Todas as calibrações foram efetuadas após um período de 60 minutos do conjunto Kinect-Projetor já estar ligado, seguindo o estudo apresentado em [31].

Para realizar uma calibração que cobrisse a maior área de projeção/visão do Kinect, foi necessário adotar o seguinte procedimento ao gerar novas imagens:

1) Rotacionar o padrão em ao menos 5 angulações de no máximo 20^o cada, como na Figura 3.1(sem rotação, rotação para a esquerda, direita, cima e baixo e/ou com algumas concatenações).

2) Uma vez com as possibilidades usadas, o padrão era movido para uma nova altura, como na Figura 3.2, e o processo se repetia.

3) Com ao menos 2 alturas exploradas, se movia o padrão na horizontal a fim de explorar da melhor maneira possivel todo o campo de visão do sistema, como visto na Figura 3.3.

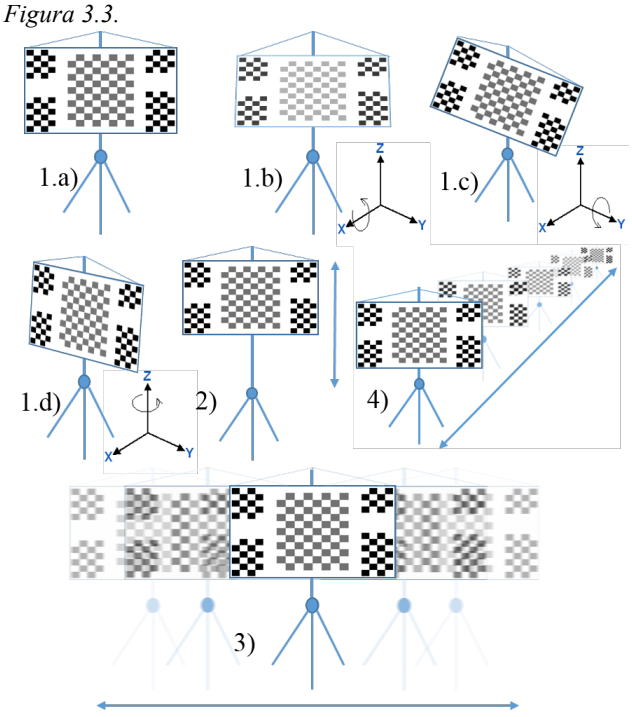

Figura 3- Diferentes posições e angulações usadas durante uma calibração.

4) Finalmente, para a calibração do sistema Kinect-Projetor, era necessário explorar também a profundidade de calibração, como na Figura 3.4, então além dos procedimentos descritos anteriormente, foi necessário se aproximar/afastar do Kinect/Projetor.

Figura 4 - Diferentes posições usadas no Sistema Kinect-Projetor

Para assegurar que o resultado da calibração era coerente, foram testados dois posicionamentos diferentes do Kinect em relação ao Projetor, conforme a Figura 4: o do Kinect e Projetor estando em posições diferentes e possuíndo uma rotação entre eles (Figura 4, esquerda) e o caso onde o Kinect estava diretamente em cima do Projetor e não havia rotação aparente entre eles (Figura 4, direita). Foi importante testar esses casos para provar a corretude dos métodos de calibração, pois o primeiro gera um sistema de equações mais complexas de serem resolvidas, logo, está mais propenso a falhas e, como se constatou que ainda assim a calibração funcionava, apesar de não ser o caso ideal, se mudou para o caso da sobreposição das câmeras para que o resultado obtido fosse melhor.

Com os estudos acima finalizados, para se atingir o resultado final desejado, foi necessário ainda pesquisar sobre métodos eficientes que apresentassem corretamente os dados oriundos do Kinect, sendo um método para visualização dos dados 2D e outro para a visualização dos dados 3D, sendo estes apresentados a seguir.

Aplicativo base NIViewer

Para auxiliar no desenvolvimento da biblioteca que fazia uso das funções oferecidas pelo OpenNI, foi utilizado como base o programa NIViewer. Este programa fornece as principais funções desenvolvidas para o uso do Kinect, como visualização da Câmera RGB e IR, além de diferentes mapas de profundidade, múltiplas visualizações e possibilidade de gravação de áudio e vídeo bem como seu gerenciamento.

A Figura 5 fornece os principais métodos de visualização fornecidos pelo NIViewer, onde todos são feitos a partir do Mapeamento de Texturas em OpenGL, fornecendo assim somente uma visualização 2D dos recursos disponíveis no Kinect. Apesar da visualização não ser feita em três dimensões, é possível notar como as imagens com Mapeamento de Profundidade apresentam a ideia de profundidade de uma maneira melhor que a fornecida pela Câmera RGB.

Na Figura 5.a), temos um Mapeamento de Profundidade feito somente com uma escala de cor, onde quanto mais contraste a cor possui, mais perto um ponto se encontra do Kinect.

(A) (B)

Figura 5 – Mapeamento de Profundidade (A) e Cores (B).

Point Cloud Library

A biblioteca Point Cloud Library (PCL) [32] oferece amplos recursos de visualização, gerenciamento e tratamento de Nuvens de Pontos acrescentando por isso diversas dependências a um projeto que só faria uso da visualização de uma Nuvem de Pontos. Visando simplificar o desenvolvimento da pesquisa, foi desenvolvido um módulo capaz de atingir resultados semelhantes ao da visualização feita pela PCL, conforme apresentado na Figura 6.

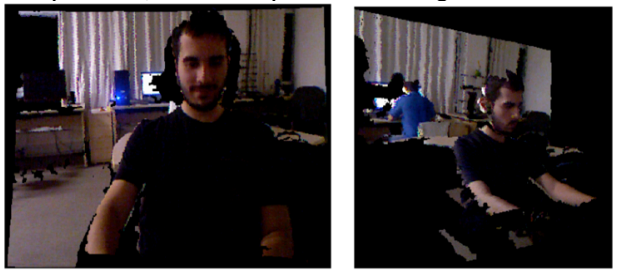

Figura 6 - Visão frontal (à esquerda) e lateral (à direita) da Nuvem de Pontos criada pelo PointCloudAssignment

V. PROJETO E ESPECIFICAÇÃO DO SISTEMA

A seguir são descritos diversos detalhes do framework desenvolvido:

A. *Arquitetura do Sistema*

1) Extração dos parâmetros intrínsecos e extrínsecos da câmera RBG do Kinect em relação à câmera IR também do Kinect:

a) Movimentação do padrão de calibração em frente às câmeras.

b) Realizar diversas snapshot do padrão de calibração em diferentes posições e angulações.

c) Processar as snapshots usando o programa de calibração RGBDemo.

2) Extração dos parâmetros intrínsecos e extrínsecos do Projetor em relação à câmera RGB do Kinect:

a) Posicionar o padrão de calibração em frente às câmeras.

b) Realizar diversas snapshot do padrão de calibração em diferentes posições e angulações.

c) Processar as snapshots usando o programa de calibração RGBDemo ou CameraProjectorCalibration.

3) Correlação entre os pontos de ambas as calibrações:

a) Realizar a transformação dos pontos no Sistema de Coordenadas da Câmera IR para o Sistema de Coordenadas do Projetor

4) Verificação da qualidade de ambas as calibrações:

a) Usar uma Nuvem de Pontos para validar as calibrações dos passos anteriores ao verificar que os objetos projetados se encontram sobre os objetos reais.

b) Caso o passo anterior falhe, recomeçar do passo 1.a)

5) Inserção de informações extras à cena real para criar uma Realidade Aumentada:

a) Criar um tratamento sobre a Nuvem de Pontos gerada no passo 4 para inserir informações virtuais na cena real.

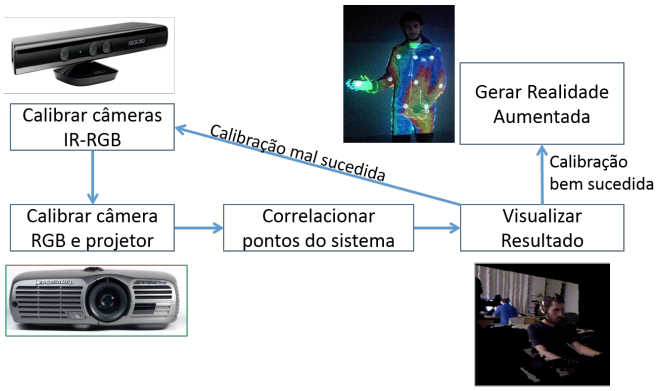

Figura 7 – Arquitetura do Sistema

B. *Características do Sistema*

O programa e todas as suas dependências foram escritas na linguagem C++ e usaram o ambiente de desenvolvimento Visual Studio 2010. Como já especificado em 4.1, o driver usado para capturar as informações do Kinect foi o OpenNI versão 1.5.4, junto com o driver Prime Sense NITE versão 1.5.2 e com o Sensor Kinect versão 5.1.2.1.
Como dependências, foi usada a biblioteca Glut versão 3.7.6 e OpenGL versão 3 para a criação e gerenciamento dos elementos gráficos. O OpenCV[33] versão 2.4.2 foi usado

para facilitar o manuseio dos dados algébricos proveniente das calibrações.

O projetor usado para testes foi um Projection Design evo2sx+, com contraste de 2500:1, 2500 ANSI Lumens, resolução 1600x1200 e throw ratio 1.78 - 2.23 : 1.

O computador usado para testes foi um laptop com processador Intel 2nd Gen Core i7-2670QM, com 4 núcleos físicos e 4 núcleos virtuais e clock de 2.20 GHz, memória RAM de 8.00GB, placa gráfica dedicada GeForce GT540M, , placa gráfica integrada Intel(R) HD Graphics 3000, HD Serial ATA-300 de 750.0 GB , com sistema operacional Microsoft Windows 7 Home Premium 64-bits Edition tela com resolução de 1600x900.

Os padrões de calibração usados foram impressos em papel sulfite com 75 de gramatura, tamanhos A4 para calibração do Kinect e A1 para a calibração do sistema Kinect-Projetor. Ambos os padrões físicos foram fixados em papel pluma e posteriormente fixados em um tripé para câmeras.

O projetor foi posicionado de cabeça para baixo para que a imagem gerada tivesse como ponto mais alto a altura do projetor e tendo seu ponto mínimo em função da distância de projeção. Essa solução foi necessária para que o Kinect pudesse capturar toda a área de projeção do projetor sem gerar oclusão e para que a projeção pudesse ser feita sobre uma pessoa, tendo como altura máxima cerca de 1.80m

C. Bibliotecas e Módulos criados

Para recuperar os dados obtidos com a calibração entre as câmeras, foi criado um módulo chamado de Calibration, que foi baseado no programa Cámara Lúcida[34]. Este módulo é responsável por carregar as matrizes de calibração no programa para que estas possam, futuramente, alterar as matrizes de Modelview e Projection do OpenGL e também por armazenar os dados intrínsecos e extrínsecos de cada câmera em seu respectivo dispositivo.

A criação das matrizes de Modelview e Projection é feita pelo módulo OpticalDevice, que resolve a álgebra necessária, explicada na seção III, para utilizar em conjunto todas as matrizes obtidas com a calibração entre as câmeras.

O módulo HelpMenu foi criado para auxiliar o usuário ao fornecer na tela as teclas de atalho necessárias para o manuseio do programa e suas respectivas tooltips.

Para que fosse possível ter visualizações diferentes da Nuvem de Pontos foi necessário criar um módulo capaz de interpretar movimentos do mouse e manusear a Nuvem de Pontos de acordo com este movimento.

D. NIViewerLib - Biblioteca para acesso das funções oferecidas pelo Kinect

Visando a reutilização, gerenciamento e manutenção dos métodos que utilizam os recursos oferecidos pelo Kinect, foi desenvolvida uma biblioteca que engloba suas principais funções, como:

1) Inicialização.

2) Obtenção e visualização dos diferentes streams fornecidos pelo OpenNI.

3) Redimensionamento do stream.

Teste no contexto OpenGL com matrizes simples de Modelview e Projection

Foi criada uma biblioteca para uso das funcionalidades oferecidas pelo OpenNI. A biblioteca criada fornece um uso modular para os métodos necessários, encapsulando módulos de acesso direto aos dados brutos do Kinect e possibilitando a visualização apenas das funções necessárias ao usuário.

Os protótipos da biblioteca OpenNI criados fornecem as mesmas funcionalidades, todas apresentadas em 2D:

1) Visualização da Câmera RGB/IR.

2) Visualização de diferentes mapas de profundidade sobre a Câmera IR.

3) Visualização da Câmera RGB com informações de profundidade.

4) Visualização lado-a-lado de ambas as câmeras.

5) Gerenciamento e visualização em 2D e 3D do Esqueleto

E. Point Cloud Viewer

Um software foi desenvolvido em OpenGL para garantir a utilização, manutenção e gerenciamento de uma Nuvem de Pontos. Ë possível criar um mapa de profundidade, com um fator de transparência, por cima da Nuvem de Pontos da Câmera RGB, fornecendo assim uma informação fácil de ser entendida para o usuário sobre a distância em que cada ponto da cena se encontra, este modo será chamado aqui de Nuvem de Pontos com Mapeamento de Profundidade. Na Figura 8 é possível observar que pontos que se encontram mais próximos do Kinect possuem um tom azulado, enquanto pontos mais afastados são esverdeados.

O software da Nuvem de Pontos com Mapeamento de Profundidade foi necessário, uma vez que ao tentar validar uma calibração com o protótipo, tínhamos apenas uma Nuvem de Pontos da Câmera RGB sendo projetada em cima de uma cena. Não sendo esse um método eficiente de validação por não ser possível visualizar se os pontos projetados de fato se sobrepõe aos pontos reais. Com a informação do mapeamento de profundidade com um fator de transparência adicionada à Nuvem de Pontos, foi mais fácil realizar esta validação.

Figura 8 - Visão frontal (à esquerda) e lateral (à direita) da Nuvem de Pontos com Mapeamento de Profundidade

VI. RESULTADOS FINAIS

O programa final desenvolvido permite ao usuário carregar os arquivos de calibração. Uma vez com os arquivos carregados, o programa se encarrega de interpretar as matrizes fornecidas e gerar as matrizes de Modelview e Projection necessárias para gerar a projeção mapeada. É importante notar que as matrizes fornecidas pelo programa de calibração estão no formato do OpenCV e por isso são orientadas por linha, precisando assim serem transformadas para o formato OpenGL, ficando orientadas por coluna.

Os diferentes resultados obtidos com o framework estão apresentados na Figura 9, Figura 10 e Figura 11. Todas as formas de mapeamento possíveis apresentadas pelo framework são feitas com o auxílio de uma nuvem de pontos e podem ser visualizadas, em tempo real, em quaisquer tipos de superfícies, sejam elas bem definidas ou não. Na Figura 9 temos um tabuleiro de xadrez que se encontra rotacionado em cerca de 30º em relação ao sistema Kinect-Projetor, sendo este exemplo uma superfície bem definida. É possível observar nesta imagem a diferença de profundidade que existe do início ao final do tabuleiro de xadrez. A escala usada para mostrar a diferença de profundidade é baseada em um ciclo de cores que vai do vermelho ao azul.

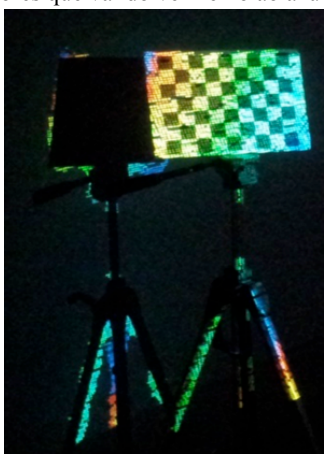

Figura 9 - Tabuleiro de Xadrez no Tripé sob Mapeamento de Profundidade com Escala Cíclica

A Figura 10 à esquerda permite a visualização de ainda mais níveis de profundidade, ressaltando com exatidão a diferença de distâncias entre os ombros, o torso e as mãos.

Repare que na Figura 11 à direita, a informação de cores obtida com a câmera RGB do Kinect se sobrepõe corretamente sobre os objetos contidos na cena. Já na Figura 11 à esquerda temos o Mapeamento de Esqueleto ocorrendo com precisão em cima de um usuário. Ambas as Figura 10 e Figura 11 são exemplos de superfícies que não são bem definidas.

Figura 10 - À esquerda: Mapeamento de Profundidade com Escala Cíclica. À direita: Mapeamento de Profundidade com Duas Escalas de Cor. Em ambos existem o mapeamento de esqueleto sobreposto aos seus respectivos mapeamentos.

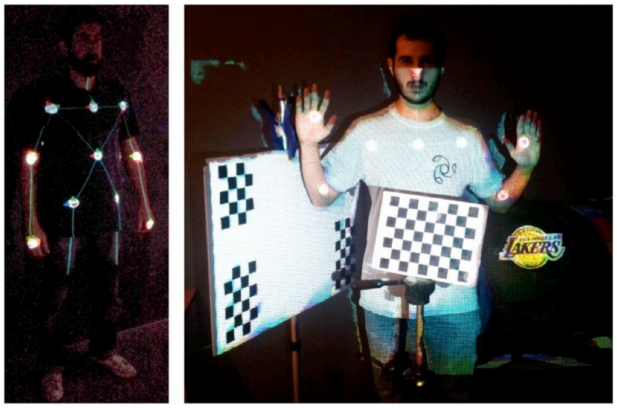

Figura 11 - À esquerda: Mapeamento de Esqueleto. À direita: Mapeamento de Profundidade com informação da câmera RGB

Quanto mais nuâncias uma cena possuir e quanto menos linear forem as bordas dos objetos contidos nela, menor é a chance de toda a sua superfície ser integralmente e corretamente iluminada e capturada pelo sistema Kinect-Projetor, acarretando assim em menos informações geradas para a cena e piorando o comportamento da Projeção Mapeada para estes objetos.

A. Dificuldades encontradas

Quase todos os problemas encontrados ao finalizar o sistema fazem parte do processo de calibração. Eles serão explicados a seguir:

1) Falta de controle de White Balance e Auto Exposure no Kinect: Para conseguir usar com sucesso o programa de calibração CameraProjectorCalibration, foi necessário usar o driver do Microsoft SDK. Entretanto, como o Kinect usado foi o da versão do XBOX 360 e não a versão do Kinect for Windows, não foi possível ter nenhum tipo de controle sobre sua câmera RGB, o que dificultou

drasticamente a calibração, uma vez que a luz emitida pelo projetor, ao ser filtrada pelo Auto Exposure, acabava diminuindo a luz ambiente geral, dificultando assim o reconhecimento do padrão físico. Quatro opções foram encontradas para contornar este problema:

a) Adicionar uma nova câmera, de alta qualidade, ao sistema e realizar a calibração dessa câmera com a câmera RGB do Kinect, adicionando assim uma complexidade a mais ao sistema.

b) Criar um driver capaz de reconhecer outros drivers do Kinect como webcam. Drivers como o libfreenect e o OpenNI versão 2.x afirmam ter os controles necessários para contornar este problema, entretanto a construção deste driver fugiria do tema proposto.

c) Realizar um refatoramento em todo o projeto de calibração a fim de retirar a dependência de input do CameraProjectorCalibration.

d) A aquisição de um Kinect for Windows. O testes foram feitos usando a versão padrão vinda com o XBOX usando um cabo adaptador usb.

2) A câmera RGB do Kinect possui uma baixa qualidade de imagem. Seria assim necessário que tanto o padrão físico quanto o virtual estivessem próximos do Kinect para que a detecção fosse realizada com sucesso. Apesar disso, como o Kinect foi posicionado acima do Projetor temos que quanto mais próximo o padrão se encontra do Kinect, menor será a projeção, dificultando mais uma vez a detecção dos padrões. Um compromisso entre qualidade de calibração e distância de calibração deve ser atingido. Por esse motivo o padrão físico foi impresso em uma folha A1 para que se pudesse cobrir uma distância maior de calibração.

3) O programa ProjectorCameraCalibration possui gargalos de desempenho na hora de realizar a calibração, fazendo assim com que fosse percebido um atraso de resposta do sistema ao se capturar uma imagem. Como o sistema é automatizado, esse atraso muitas vezes significava uma captura de imagem feita ao mesmo tempo em que o padrão estava sendo movido, fazendo assim com que a imagem ficasse borrada e resultando em uma má qualidade de calibração. Não foi encontrado o motivo desse gargalo, portanto o método encontrado para minimizá-lo era aumentar o intervalo com que as imagens eram capturadas, aumentando consideravelmente o tempo total de calibração.

4) O tempo para realizar uma calibração sobre qualquer um dos métodos vistos era muito superior aos encontrados em calibrações de câmeras de maior qualidade.

5) Como o programa de visualização usa o driver do OpenNI e o programa de calibração usa o driver do Microsoft Kinect SDK, é necessário ter ambos os drivers instalados no computador e fazer a troca entre eles quando necessário, o que aumenta a dependência do processo como um todo para se obter um bom resultado.

6) Na Figura 10 é possível reparar que a projeção extrapola um pouco a área do corpo. Isso acontece pelos seguintes motivos:

a) Como a visualização é feita com uma nuvem de pontos, cada pixel da visualização é um ponto desenhado pelo OpenGL, entretanto o desenho de um ponto pelo OpenGL possui um tamanho mínimo, que é maior que o necessário para se preencher corretamente toda a área.

b) Quando esta imagem foi feita, o tamanho do pixel estava definido como 3, porém a mudança desse tamanho para 1 fazia com que a nuvem de pontos ficasse mais esparsa, prejudicando a visualização.

c) Cada ponto da nuvem de pontos é fornecido pelo Kinect, então para evitar que a nuvem de pontos fique esparsa com um tamanho menor de ponto no OpenGL, seria necessário que o projetor de IR do Kinect pudesse projetar mais pontos.

d) Erro de reprojeção: Um dos parâmetros para verificar se a calibração se deu corretamente é o valor do erro de reprojeção, quanto maior ele é, menos acurada foi a calibração feita e menor será a sobreposição entre o real e o virtual. Como para realizar a projeção mapeada é necessário efetuar duas calibrações, o erro de reprojeção final será maior que o erro isolado de cada uma das calibrações.

VII. CONCLUSÕES

Este trabalho buscou apresentar um framework capaz de ler uma calibração feita em um sistema Kinect-Projetor e validá-la. A validação é feita ao apresentar diferentes formas de visualização dos dados oriundos de um Kinect. Este é um interesse tanto de desenvolvedores quanto de usuários que tenham interesse em Realidade Aumentada.

Desenvolvedores serão capazes de conferir seus métodos de calibração usando este framework, além de poderem estudar a reação que os métodos de visualização aqui usados causam nas pessoas, podendo assim criar novas formas de apresentação de dados.

Devido às limitações encontradas no Kinect e nos métodos de calibração, há bastante espaço para trabalhos futuros, vistos a seguir:

Uso do Kinect 2.0, que possui uma resolução melhor (1080p) e mais recursos que o Kinect 1.0, prometendo assim solucionar vários dos problemas aqui encontrados.

Métodos mais eficientes de calibração ainda precisam ser estudados para que se consiga atingir uma melhor visualização da Projeção Mapeada e em menos tempo.

Visto que o Kinect 2.0 não possui mais parceria com a Primesense, seria prudente usar o driver da Microsoft ao invés do driver do OpenNI para assim garantir que o framework funcione com ambas as versões do Kinect ou que a manutenção necessária para uso de ambas seja a menor possível.

Testar padrões diferentes de calibração, visando assim atingir um menor erro de reprojeção.

Atualmente o tamanho do padrão de calibração usado impõe uma restrição à este trabalho, tanto por ser difícil de manuseá-lo quanto pela dificuldade em achar uma superfície

leve e rígida o suficiente para não causar dobras ou distorções no padrão. É necessário procurar por padrões menores e algoritmos mais eficientes que consigam detectá-los.

AGRADECIMENTOS

Os autores gostariam de agradecer o Tecgraf/PUC-Rio por todo o apoio a esse projeto. Luciano Soares foi financiado pelo CNPq, processo 483195/2011-1.

REFERÊNCIAS

[1] R. T. Azuma, "A Survey of Augmented Reality," Presence - Teleoperators and Virtual Environments, vol. 6, pp. 355-385, 1997.

[2] Pinto, F., Buaes, A., Francio, D., Binotto, A. P. D., & Santos, P. (2008, August). BraTrack: a low-cost marker-based optical stereo tracking system. In SIGGRAPH Posters (p. 131).

[3] ART – AdvancedRealtimeTracking - http://www.ar-tracking.com/home/ - Visitado pela última vez em 22/11/2013

[4] Nova3D – Projeção Mapeada - http://www.nova3d.com.br/#!projecao-mapeada.html - Visitado pela última vez em 21/11/2013

[5] Bot& Dolly - http://www.botndolly.com/box - Visitado pela última vez em 21/11/2013

[6] Bimber, O., Raskar, R., &Inami, M. (2005). Spatial augmented reality. Wellesley: AK Peters.

[7] UAU Mídia Interativa - http://uaumidia.com.br/projecaomapeada - Visitado pela última vez em 21/11/2013

[8] LPMT - http://hv-a.com/lpmt/ - Visitado pela última vez em 21/11/2013

[9] CarProjection - http://carprojection.com/ - Visitado pela última vez em 21/11/2013

[10] A Dandy Punk - http://www.adandypunk.com/#!viddywell/c1t44 - Visitado pela última vez em 21/11/2013

[11] ARPool - http://arpool.ca/ - Visitado pela última vez em 21/11/2013

[12] Ziola, R., Grampurohit, S., Nate, L., Fogarty, J., Harrison, B. OASIS: Creating Smart Objects with Dynamic Digital Behavior,Workshop at IUI 2011.

[13] Kondo, D., Goto, T., Kouno, M., Kijima, R., & Takahashi, Y. (2004). A virtual anatomical torso for medical education using free form image projection. InProceedings of 10th International Conference on Virtual Systems and MultiMedia (VSMM2004) (pp. 678-685).

[14] RGBDemo - http://labs.manctl.com/rgbdemo/ - Visitado pela última vez em 22/11/2013

[15] Audet, S. (2012). Markerless Interactive Augmented Reality on Moving Planar Surfaces with Video Projection and a Color Camera (Phd. Tokyo Institute of Technology).

[16] Jones, B., Sodhi, R. (2010, december). OpenLight - Kinect-Projector Calibration Disponible at http://augmentedengineering.com/ProCamCalibration/brjones2rsodhi2Final.pdf - Visitado pela última vez em 22/11/2013

[17] CameraProjectorCalibration - https://github.com/alvarohub/Example_CameraProjectorCalibration - Visitado pela última vez em 22/11/2013

[18] Raskar, R., Baar, J., Beardsley, P., Willwacher, T., Rao, S. and Forlines, C. 2003. iLamps: geometrically aware and self-configuring projectors. ACM Trans. Graph. 22, 3 (July 2003), 809-818.

[19] Chung-Sayers, N. "TEAM USES *XBOX *KINECT TO SEE PATIENT IMAGES DURING SURGERY", Sunnybrook report, Toronto, ON (March 14, 2011)

[20] IHS 2011 http://intelligenthealingspaces.com/

[21] Zhang, Z. (1998), A flexible new technique for camera calibration, Technical report, Microsoft Corporation.

[22] Tsai , R. Y. (1987), 'A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses', Ieee Journal Of Robotics And Automation RA-3(4), 323–344.

[23] Izadi, S.; Kim, D.; Hilliges, O.; Molyneaux, D.; Newcombe, R.; Kohli, P.; Shotton, J.; Hodges, S.; Freeman, D.; Davison, A.; et al. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In Proceedings of ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, 16–19 October

[24] PrimeSense Technology - http://www.primesense.com/solutions/technology/ – Visitado pela última vez em 22/11/2013

[25] OpenNI - http://www.openni.org/ - Visitado pela última vez em 22/11/2013

[26] Kinect for Windows SDK - http://www.microsoft.com/en-us/kinectforwindowsdev/Start.aspx - Visitado pela última vez em 22/11/2013

[27] OpenKinect Project - http://openkinect.org/wiki/Main_Page - Visitado pela última vez em 22/11/2013

[28] Oren Ben-Kiki, Clark Evans und Ingy döt Net. YAML Ain't Markup Language (YAML). 2009. url: http://yaml.org/spec/1.2/spec.pdf (besucht am 22. 07. 2012).

[29] OpenFrameworks - http://www.openframeworks.cc/ - Visitado pela última vez em 22/11/2013

[30] Kinect for PC and Skype meansKinectCam - http://piotrsowa.eu/2013/03/17/kinect-for-pc-and-skype-means-kinectcam/ – Visitado pela última vez em 22/11/2013

[31] Fiedler, D., & Müller, H. (2012, November). Impact of thermal and environmental conditions on the kinect sensor. In Proc. Int. Workshop onDepthImageAnalysis.

[32] Aldoma, A., Marton, Z. C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., ... & Vincze, M. (2012). Point Cloud Library. IEEE Robotics & Automation Magazine, 1070(9932/12).

[33] OpenCV, L. (2008). Computer vision with the OpenCV library. GaryBradski & Adrian Kaebler-O'Reilly.

[34] Cámara Lúcida. R+D - http://www.camara-lucida.com.ar/ – Visitado pela última vez em 22/11/2013