REGISTRO E ALINHAMENTO DE IMAGENS DE PROFUNDIDADE OBTIDAS COM DIGITALIZADOR PARA O MODELAMENTO DE OBJETOS COM ANÁLISE EXPERIMENTAL DO ALGORITMO *ICP*

JOSE ALEJANDRO BONILLA NARANJO

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

DEPARTAMENTO DE ENGENHARIA MECÂNICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

REGISTRO E ALINHAMENTO DE IMAGENS DE PROFUNDIDADE OBTIDAS COM DIGITALIZADOR PARA O MODELAMENTO DE OBJETOSCOM ANÁLISE EXPERIMENTAL DO ALGORITMO *ICP*

JOSE ALEJANDRO BONILLA NARANJO

ORIENTADOR: JOSÉ MAURICIO TORRES SANTOS DA MOTTA

DISSERTAÇÃO DE MESTRADO EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO:

BRASÍLIA/DF: OUTUBRO – 2012

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

REGISTRO E ALINHAMENTO DE IMAGENS DE PROFUNDIDADE OBTIDAS COM DIGITALIZADOR PARA O MODELAMENTO DE OBJETOS COM ANÁLISE EXPERIMENTAL DO ALGORITMO *ICP*

JOSE ALEJANDRO BONILLA NARANJO

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

Prof. José Mauricio Santos Torres da Motta, PhD. (ENM-UnB)

(Orientador)

(Examinador Interno)

(Examinador Externo)

BRASÍLIA/DF

FICHA CATALOGRÁFICA

NARANJO, JOSE ALEJANDRO BONILLA

Registro e alinhamento de imagens de profundidade obtidas com digitalizador para o modelamento de objetos com análise experimental do algoritmo ICP [Distrito Federal] 2011.

xv, 108p., 210 x 297 mm (ENM/FT/UnB, Mestre, Sistemas Mecatrônicos, 2012). Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Mecânica.

1. Registro de imagens de profundidade
2. Modelamento de objetos em 3-D
3. Algoritmo *ICP*
4. Algoritmo *kd-tree*

I. ENM/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

NARANJO, J. A. B. (2012). Registro e alinhamento de imagens de profundidade obtidas com digitalizador para o modelamento de objetos com análise experimental do algoritmo *ICP*. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação ENM.DM-55A/12, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 108p.

CESSÃO DE DEREITOS

AUTOR: Jose Alejandro Bonilla Naranjo
TÍTULO: Registro e Alinhamento de Imagens de Profundidade Obtidas com Digitalizador para o Modelamento de Objetos com Análise Experimental do Algoritmo *ICP*

GRAU: Mestre ANO: 2012

Jose Alejandro Bonilla Naranjo
SCLN 207 Bloco C, AP 107
4.613.890 – Brasília – DF – Brasil.

RESUMO

REGISTRO E ALINHAMENTO DE IMAGENS DE PROFUNDIDADE OBTIDAS COM DIGITALIZADOR PARA O MODELAMENTO DE OBJETOS UTILIZANDO O ALGORITMO *ICP*

Este trabalho tem o foco na utilização de técnicas de visão computacional para o desenvolvimento de algoritmos de registro, alinhamento e modelamento de objetos em 3-D. O registro de imagens é realizado utilizando duas metodologias no algoritmo *ICP* (*Iterative Closest Point*). Na sua etapa de "Busca de correspondências" a primeira metodologia, chamada "força bruta", encontra os pontos correspondentes com base nas distâncias entre os pontos da nuvem de pontos base com relação à nuvem de pontos modelo, e a segunda, chamada "*kd tree*", acelera a busca do ponto mais próximo entre duas nuvens de pontos.

Além disso, é construído um algoritmo que utiliza múltiplas imagens de profundidade para realizar o modelamento 3-D de um objeto utilizando o algoritmo *ICP*. Este algoritmo requer conhecimento do sistema estudado para realizar o alinhamento prévio das nuvens de pontos e, posteriormente, efetuar o registro de cada uma das nuvens de pontos do mesmo objeto em diferentes posições para reconstrui-lo digitalmente.

A convergência do algoritmo *ICP* é determinada utilizando o "Erro Quadrático Médio" (*Root Mean Square - RMS*), é um parâmetro que serve como critério de parada e medição da convergência no registro em cada iteração.

O principal aporte desta pesquisa foi oferecer um uso diferente do algoritmo *ICP* para o modelamento de objetos. Não obstante seja preciso reconhecer que o método desenvolvido neste trabalho é um pouco primitivo no sentido de que ainda depende em muitas etapas da intervenção humana. É o ponto de partida para diversas pesquisas futuras no campo do modelamento de objetos, que atualmente, é considerado a "obra-prima" da reconstrução de modelos, já que oferece o nível mais alto de desenvolvimento na invenção de técnicas e algoritmos.

ABSTRACT

REGISTRATION AND ALIGNMENT OF DEPTH IMAGES, SCANNER OBTEINED, FOR OBJECT MODELING, USING THE ICP ALGORITHM

This work focuses on the use of computer vision techniques to develop registration algorithms, alignment and 3D object modeling. The image registration process is performed using two methodologies for the ICP algorithm (Iterative Closest Point) during its "Matching" stage. The first methodology, called "brute force", finds the corresponding points based on the distances between the points from the “base" point cloud and the "Model" point cloud, the second one, called "KD Tree", accelerates the search of the nearest point between two point clouds. Furthermore, another algorithm is built to perform the object modeling task, which requires knowledge of the system to perform a preliminary alignment of the point clouds to proceed to the registration of a pair of multiple point clouds of the same object from different positions and to digitally reconstruct it.

Furthermore, an algorithm is built which uses multiple depth images to perform the 3D modeling of an object using the ICP algorithm. This algorithm requires knowledge of the studied system to perform a previous alignment of the point clouds and, subsequently, perform the registration of each of the point clouds of the same object in different positions to digitally reconstruct it.

The convergence of the algorithm ICP is determined using the Root Mean Square, although not an enough condition to ensure the convergence of the algorithm, this is a parameter that serves as a stop criterion and convergence measurement at each iteration record.

The main contribution of this research is to offer a different use for the ICP algorithm for object modeling. Nevertheless, it must be recognized that the method developed in this work is primitive in the sense that it still depends on many stages of human intervention. Thus this is the starting point for future research in the field of object modeling, currently considered the "masterpiece" of model reconstruction, given that it offers the highest level of development in the invention of techniques and algorithms.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS.

LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

AGs - *Genetic Algorithm.*

BSP - *Binary Space Partitioning.*

CCD - *Charge – Coupled Device.*

DARCES - *Data Aligned Rigidity Constrained Exhausive Search.*

ICP - *Iterative Closest Point.*

Kd tree- *k-Dimensional tree.*

NPBS - *N-planes B - Spline.*

.ply - *Polygon File Format.*

RMS - *Root Mean Square.*

SVD - *Singular Value Decomposition.*

.txt - *Text File.*

3D - *Three - dimensional space.*

VFSR - *Very Fast Simulated Reannealing.*

$\overline{q}$ - *Centróides da nuvem de pontos base.*

$\overline{p}$ - *Centróide da nuvem de pontos e modelo.*

d_s - *Distância do ponto ao plano.*

$F(p_i)$ - *Função do plano.*

ℓ_i - *Linha normal a P em* p_i.

e - *Métrica de erro.*

R - *Matriz de Rotação.*

n_i - *Normal. tangente estimada no i-esimo ponto da nuvem de pontos modelo.*

n_m - *Número de pontos da nuvem de pontos base.*

n_b - *Número de pontos da nuvem de pontos modelo.*

Q - *Nuvem de pontos base.*

P - *Nuvem de pontos modelo.*

S^k_j - *Plano tangente ao ponto* q'^k_j *da nuvem de pontos Q.*

S_j - *Plano tangente de Q em* q_j.

q_i - *Ponto da nuvem de pontos base.*

p_i - *Ponto da nuvem de pontos modelo.*

q^k_j - *Ponto de interseção entre Q com a linha* $T^{k-1}L_i$.

n_{pi} - *Superfície normal de P em* p_i.

T - *Vetor de Translação.*

n^k_{qj} - *Vetor normal de* q'^k_j *na superfície Q.*

1. INTRODUÇÃO

As técnicas de visão computacional têm sido aplicadas no avanço das tecnologias nas últimas décadas para facilitar a realização de múltiplas tarefas que antes faziam parte somente da ficção científica. Não por acaso as grandes empresas multinacionais de computação estão na vanguarda no uso das melhores tecnologias dentro dos seus dispositivos. O desenvolvimento no campo das telecomunicações permitiu a incursão de novas tecnologias que foram responsáveis por desenvolver dia a dia dispositivos mais robustos e com melhores rendimentos, também aproveitados para a evolução de dispositivos e software nos sistemas que utilizam técnicas da visão computacional. Todos esses avanços marcaram a história ao melhorar diversas áreas do conhecimento como a medicina, robótica, aplicações militares, sistemas de navegação, entretenimento e indústria em geral, para de uma forma ou outra melhorar nossas condições de vida.

A maneira como os nossos corpos registram a informação proveniente do meio ambiente depende de nossos sentidos. O cérebro opera em conjunto com os sentidos para resolver problemas como reconhecimento, identificação e detecção das características de nosso entorno, tarefa que não é nada fácil de ser reproduzida, inclusive pelos dispositivos com maior tecnologia inventados pelo homem. Por exemplo, os "*scanners*" (ou digitalizadores) de luz estruturada são dispositivos capazes de obter uma grande quantidade de dados sem efetuar contato sobre os objetos digitalizados, mas carecem da propriedade de interpretar e analisar a informação que geram (Georgopoulos, et. al., 2010).

Graças aos avanços da tecnologia, hoje é possível encontrar diferentes tipos de dispositivos tecnológicos dentro de nossa rotina diária, já que alguns vêm implantados dentro de câmeras ou de videojogos de última geração. Mas sem ter a mesma precisão de dispositivos especializados (como é o caso dos digitalizadores de precisão, destinados a cumprir o objetivo de criar apenas nuvens de pontos com a melhor precisão do ambiente recriado) existem digitalizadores a laser dentro de produtos que são vendidos nas lojas para crianças, como é o caso dos videojogos atuais.

Devido a essa expansão da tecnologia é possível, atualmente, com a ajuda de computadores, realizar estudos com foco dentro da área da visão computacional, como o registro de imagens e o reconhecimento e detecção de objetos utilizando recursos

econômicos e de fácil acesso.

Muitos trabalhos foram desenvolvidos para resolver o problema do registro de imagens em *3-D* sobre nuvens de pontos, começando com o uso de técnicas como “Rotulação Estocástica Hierárquica”, “Transformada de *Hough*”, “Quaternions”, “SVD”, etc., dentro de algoritmos iterativos baseados em pontos ou planos ou em linhas (Horn, 1987). Somente após o desenvolvimento de algoritmos robustos foram realizados avanços importantes dentro das técnicas da visão computacional para resolver o problema do registro e reconhecimento de objetos (Rusinkiewicz e Levoy, 2001). O *ICP* é um algoritmo iterativo cujas siglas significam “*ITERATIVE CLOSEST POINT*” e, como seu nome indica, estabelece uma relação entre as distâncias entre pontos dentro das imagens onde se realiza tal registro (Besl e Mckay, 1992). Depois disso muitos pesquisadores começaram a propor melhorias sobre o *ICP* essencialmente para diminuir o custo computacional do algoritmo. Os métodos para aperfeiçoar o algoritmo *ICP* podem ser separados em 3 tipos:

- Redução do número de iterações.

- Redução do número de dados (*data points*).

- Aceleração da busca dos pontos mais próximos.

O algoritmo funciona desenvolvendo uma série de procedimentos para encontrar os pontos correspondentes mais próximos entre duas imagens formadas por nuvens de pontos. As nuvens de pontos são um conjunto de coordenadas que representam a superfície externa de um objeto real dentro de um sistema de coordenadas tridimensional. Os pontos se identificam em geral como coordenadas X, Y e Z, e representam a superfície de um objeto (Trucco, e Verri, 1998).

Neste trabalho foram realizados o registro, o alinhamento e o modelamento de objetos *3-D*, representados com nuvens de pontos, utilizando duas variantes do mesmo algoritmo, um algoritmo *ICP* convencional (chamado assim porque utiliza um método de busca chamado “força bruta”) e outro algoritmo *ICP* que implementa uma melhora para diminuir o tempo para encontrar os pontos mais próximos, utilizando um algoritmo de busca chamado “*kd tree*” (Bentley, 1980). Por ultimo, é feito um modelamento de objetos utilizando um alinhamento prévio do objeto com os parâmetros extrínsecos do sistema.

1.1. JUSTIFICATIVA

A constante necessidade de melhorar as técnicas da visão computacional, com a ajuda de algoritmos de custo computacional baixo e de melhor desempenho, faz com que as diversas técnicas de reconhecimento e registro de imagens *3-D* melhorem dia após dia.

A aquisição de modelos digitais em áreas como a medicina, arqueologia, arquitetura, etc. se faz cada vez mais importante no mundo, já que em cada uma dessas áreas existem aplicações (utilizando técnicas de visão computacional apropriadas) que conseguem a maior eficiência em cada um dos processos.

Diversas aplicações, como a navegação robótica, utilizam o algoritmo *ICP* para criar robôs que executem tarefas de alto nível de autonomia para se movimentar em ambientes desconhecidos utilizando o registro de imagens. É por esse motivo que estudos nesta área, pouco a pouco, geraram avanços na tecnologia, devido que atualmente é possível encontrar sistemas que precisam de técnicas de visão computacional (como as técnicas de registro e modelamento) para realizar funções que requeiram altíssima precisão. Além disso, o registro de imagens faz parte, atualmente, do estado-da-arte dos sistemas de reconstrução digital, e, como tal, é de grande importância no desenvolvimento de tecnologias de ponta, de imensa utilidade nos sistemas mecatrônicos e contribuindo no aperfeiçoamento na área da visão computacional.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

O objetivo geral do projeto é realizar um estudo do registro e alinhamento de nuvens de pontos para o modelamento de objetos *3-D* (imagens *3-D*) aplicando o algoritmo "*ICP*", em conjunto com um algoritmo de busca denominado "kd tree", com a quantificação de seu desempenho.

1.2.2 OBJETIVOS ESPECÍFICOS

- Estudar as diversas técnicas que envolvem o algoritmo *ICP* e revisar os aspectos relacionados com as técnicas de visão computacional na área do registro e modelamento de objetos em *3-D*;

- Desenvolver o processamento de imagens *3-D* (nuvens de pontos) a partir de imagens obtidas por um digitalizador de luz estruturada;

- Construir o algoritmo *ICP* para realizar o registro e alinhamento de objetos *3-D*, utilizando nuvens de pontos obtidas com um digitalizador de luz estruturada;

- Implementar uma melhora do algoritmo "*ICP*" utilizando um algoritmo de busca chamado "*kd tree*" e analisar o custo computacional do algoritmo resultante;

- Provar a convergência do algoritmo *ICP* em presença de dados discrepantes, utilizando diversas nuvens de pontos tanto sobre o algoritmo convencional como no algoritmo melhorado com o algoritmo de busca "*kd tree*";

- Analisar os resultados obtidos com o algoritmo em presença de imagens alteradas com ruído;

- Implementar uma técnica para realizar o modelamento de objetos *3-D*, utilizando as vantagens do algoritmo *ICP*.

1.3 ESTRUTURA DO DOCUMENTO

Com o fim de alcançar os melhores resultados, foram propostos os seguintes passos no trabalho:

- Obtenção de nuvens de pontos utilizando um digitalizador de luz estruturada.

- Localização e estudo das nuvens de pontos e a sua relação espacial dentro do sistema.

- Implementação de operações e cálculos matriciais, para o armazenamento e manipulação de dados *3-D.*

- Determinação dos parâmetros extrínsecos entre o sistema de coordenadas de referência da

câmera e o sistema de coordenadas de referência do ambiente.

- Determinação do tipo de imagens analisadas e a sua dificuldade no processo de registro.

- Execução dos algoritmos (*ICP*) sobre um objeto para realizar o registro das nuvens de pontos.

- Avaliação da convergência utilizando o "Erro Quadrático Médio" para cada um dos algoritmos implementados.

- Comparação entre os algoritmos utilizados com base em suas características.

- Postulação e estabelecimento do problema da digitalização de um objeto em *3-D*.

- Implementação de uma técnica para o modelamento de objetos *3-D*, utilizando o registro de nuvens de pontos.

Os sete capítulos deste trabalho foram organizados para efetuar uma explicação progressiva da execução de cada um dos passos necessários para obter cada um dos objetivos propostos, deixando o Capítulo 1 para fazer uma introdução, propor uma justificativa, declarar os objetivos e oferecer uma estrutura metodológica do projeto, e o Capítulo 7 para registrar as referências bibliográficas.

Como a construção do algoritmo *ICP* é fundamental para alcançar o objetivo principal deste trabalho, o Capítulo 2 se concentrou em fazer o estudo da literatura especializada dos processos experimentais que têm sido feitos sobre esse algoritmo. No Capítulo 3 serão definidos diversos conceitos que são vitais para entender a maioria dos processos realizados neste trabalho, além da descrição do funcionamento do algoritmo *ICP*. No Capítulo 4 são apresentados os resultados do processo de registro de nuvens de pontos obtidos com os algoritmos mencionados no Capítulo 3, para posteriormente oferecer uma aplicação dos resultados no Capítulo 5, empregando as técnicas aprendidas (e desenvolvidas) no modelamento de objetos *3-D*. O encerramento do trabalho é feito no Capítulo 6 descrevendo as conclusões e os trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

2.1 INTRODUÇÃO AOS MÉTODOS DE REGISTRO

As imagens representadas em um computador são chamadas de imagens digitais. Uma imagem digital é uma matriz bidimensional de números composta de pequenos elementos que correspondem à menor unidade da imagem. Estes pequenos elementos são chamados píxeis como aglutinação de "***Pic****ture* e ***El****ement*", sendo "*Pix*" a abreviatura em inglês para "*Picture*". O *pixel* é o menor elemento na interface de saída de um dispositivo de exibição como uma tela ou um monitor. O conjunto de todos os *pixels* forma a imagem inteira em uma imagem digital (Trucco e Verri, 1998).

Existem dois tipos de imagens digitais dependendo a forma como o conjunto de *pixels* é armazenado (Boykov e Kolmogorov, 2004). Se a informação é registrada, dependendo da intensidade luminosa, as imagens são chamadas de imagens de intensidade, mas se a informação é registrada com base em medições de distâncias, as imagens são chamadas imagens de profundidade.

O registro de imagens é o processo de transformar diferentes conjuntos de dados, cada um com um sistema de coordenadas próprio, para um sistema de coordenadas comum e único, sendo estes conjuntos de dados definidos por imagens de intensidade ou imagens de profundidade (Nkanza, 2005). O registro de imagens também é chamado de alinhamento e tem um papel importante em áreas como a aquisição de modelos *3-D*, o reconhecimento de objetos e o processamento geométrico (Mitra, et al., 2004).

O processo de transformação consiste em determinar uma matriz de transformação ótima que permita o alinhamento entre os conjuntos de dados ou imagens espacialmente. Para determinar a matriz de transformação entre duas imagens do mesmo objeto com diferente sistema de coordenadas (como mostra a Figura 2.1), é preciso aplicar várias transformações sobre uma das imagens e comparar o resultado sobre a imagem padrão até alcançar um alinhamento entre as duas imagens, como mostra a Figura 2.2 (Gu, 1997).

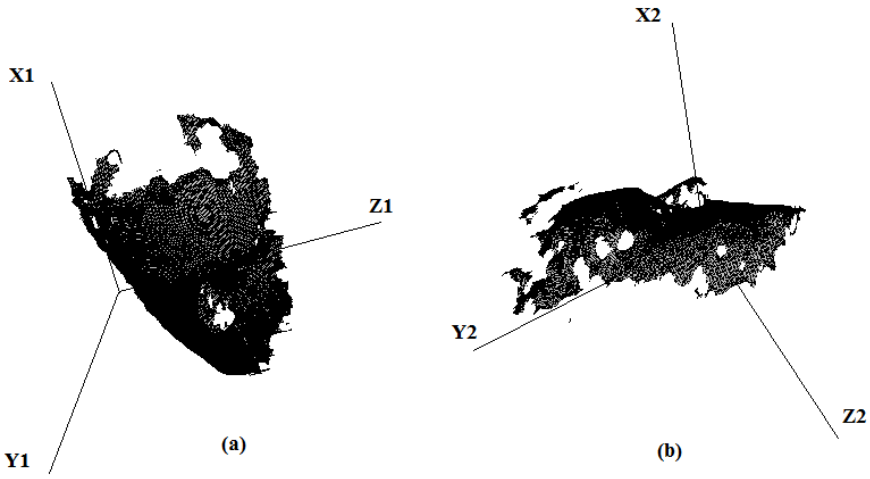


Figura 2.1 Duas imagens de um mesmo objeto com diferentes sistemas de coordenadas. (a) Objeto com sistemas de coordenadas X1, Y1, Z1. (b) Objeto com sistemas de coordenadas X2, Y2, Z2.

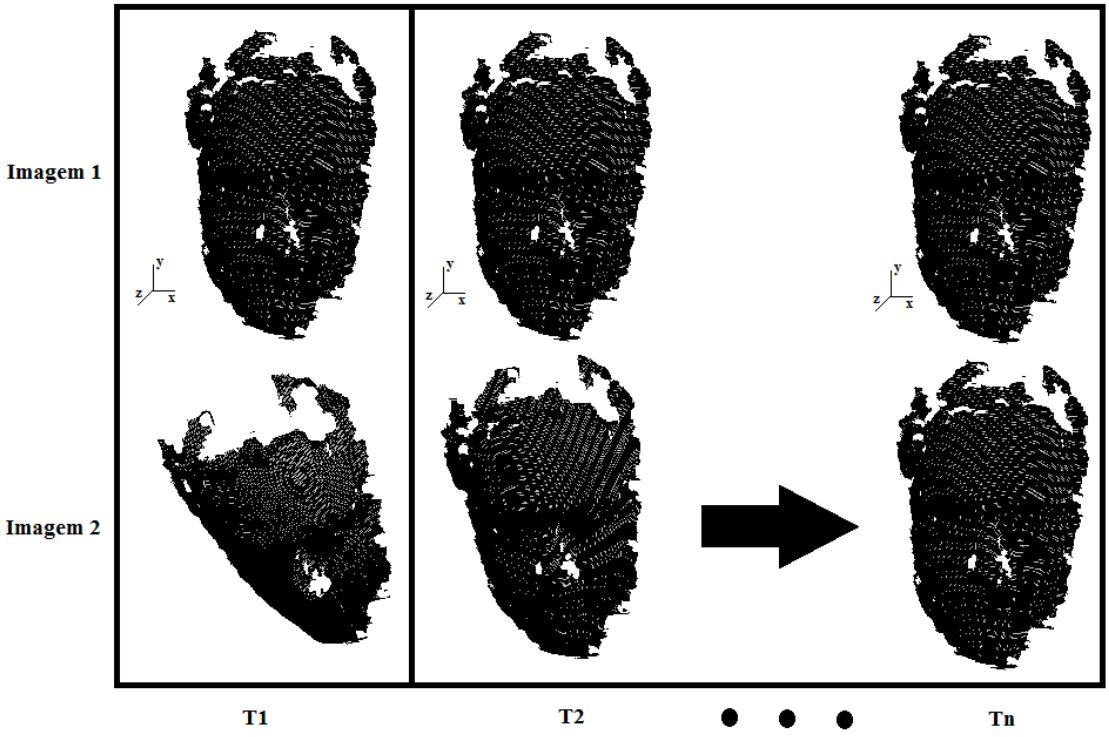


Figura 2.2 Aplicação de várias transformações (T1, T2..., Tn) até alcançar o alinhamento entre as duas imagens.

Este processo pode ser feito até alcançar um alinhamento ótimo entre duas imagens dependendo do critério do observador.

2.2 EVOLUÇÃO DOS MÉTODOS DE REGISTRO

Com a aplicação do registro e reconstrução de imagens *3-D* em muitas áreas, a demanda de novas soluções aplicadas aumentou nos últimos 70 anos. As primeiras soluções para este problema foram propostas na década de 1950, quando se resolveu o problema da orientação absoluta em aquisição estéreo como solução no campo da fotogrametria, onde uma orientação absoluta implica a determinação da rotação do modelo espacial e o problema é reduzido na solução de uma equação linear de três variáveis (Thompson, 1958) convertendo-se na solução pioneira de muitos outros tipos de soluções que seriam propostas posteriormente.

Os tipos de soluções dependem de múltiplos fatores definidos para cada aplicação específica. Dependendo da quantidade de informação que se tenha do sistema a digitalizar, os algoritmos desenvolvidos teriam mais parâmetros para procurar o registro desejado. Se a aplicação necessita de precisão nas propriedades das superfícies digitalizadas, como é o caso da reconstrução de ambientes utilizada na arquitetura, os algoritmos desenvolvidos para cumprir tal função seriam diferentes dos algoritmos desenvolvidos para aplicações em medicina, em que o fundamental é a velocidade de registro do modelo (Capel e Zisserman, 2003).

Os métodos de registro na literatura possuem diferenças na forma de resolver diferentes problemas dentro dos algoritmos, alguns deles podem realizar um alinhamento inicial que permite obter outro tipo de informações valiosas do sistema antes de realizar o registro (Salvi, et al., 2007). Por outro lado, os métodos de registro variam também na forma como realizam os processos ou etapas posteriores ao ingresso dos dados no sistema, por exemplo, a busca dos pares correspondentes ou a forma de encontrar a transformação de coordenadas entre duas imagens (Rusinkiewicz e Levoy, 2001).

O registro de imagens e a reconstrução de objetos utilizando nuvens de pontos também é uma técnica desenvolvida por muitos pesquisadores ao redor do mundo. Existem muitos trabalhos na literatura cujo objetivo é melhorar os sistemas que reconstroem modelos computacionais, seja utilizando outras técnicas computacionais, seja melhorando o tempo de execução dos algoritmos (Venâncio, 2008). Um exemplo deste tipo de algoritmos é o *ICP*.

O *ICP* é um algoritmo que oferece um método de registro buscando uma transformação

rígida para encontrar um alinhamento entre duas imagens com um processo iterativo alternando dois passos importantes: a busca pela identificação dos pontos mais próximos e a busca pela transformação rígida (Hähnelet al., 2003).

Paul Besl e Neil Mckay (1992) estabeleceram o termo *ICP* (*Iterative Closest Point*) na literatura, com ajuda de outros pesquisadores como Guillaume Champleboux (1992), que propôs uma técnica baseada na conjunção de um modelo matemático de câmera "*N-planes B-Spline*" (*NPBS*), além de desenvolver um algoritmo para a recuperação da transformação rígida entre dois conjuntos de pontos obtidos com um sensor de imagens de profundidade testado no alinhamento de superfícies de rostos humanos, obtendo resultados altamente precisos. Chia Menq, et al., (1992), determinaram a medida entre os dois conjuntos de pontos minimizando a soma das distâncias quadráticas, dos pontos correspondentes, utilizando os parâmetros de transformação de corpo rígido, para demonstrar o resultado estatístico da quantidade mínima de pontos medidos e a importância de usar algoritmos de correspondência ótimos na medida do perfil de superfícies com precisão. Yang Chen e Gerard Medioni (1992) mudaram o algoritmo aplicando uma forma diferente de realizar as correspondências utilizando uma heurística de combinação chamada "ponto ao plano". Zhengyou Zhang (1992), com um sistema de detecção de bordas baseado em visão estéreo, efetuou uma combinação entre curvas de forma livre, minimizando a função de custo para reduzir a distância media entre os pontos das imagens no percurso de um automóvel e calcular a distância total percorrida.

Até aqui, os métodos propostos para aperfeiçoar o algoritmo *ICP* não aproveitavam ao máximo as condições externas (como a localização no ambiente). A partir daí, começaram a surgir trabalhos aproveitando características da localização do objeto no ambiente de estudo, mas conservando o mesmo conceito de minimização iterativa da função de custo ou do erro. Trabalhos, por exemplo, o estudo feito por Gerard Blais e Martin Levine (1995), que utilizaram um dispositivo devidamente calibrado registrando, assim, imagens de profundidade para a construção de modelos de superfície de objetos em 3-D. Eles (Blais e Levine, 1995) encontraram os parâmetros de translação e rotação com imagens de profundidade não superpostas e reconstruindo as superfícies parciais que representavam um objeto mediante a minimização da função de custo aplicando uma técnica de otimização estocástica chamada "*very fast simulated reannealing*" (*VFSR*).

Outro trabalho interessante é descrito no livro de Luciano Silva e Olga Bellon (1995). Eles

definiram o conceito de “medida de interpenetração de superfície” e junto com Algoritmos Genéticos (*GAs*) fizeram o registro de várias vistas sem precisar obter os parâmetros de translação e rotação previamente, como era a mecânica do algoritmo *ICP*.

Outro método para o registro foi o “*data aligned rigidity constrained exhaustive search*” mais conhecido como “*DARCES*” (Chen et al., 1999). Este método realiza uma busca exaustiva dos parâmetros de rotação e translação, utilizando no mínimo três pontos para encontrar os parâmetros que permitiriam fazer o registro de imagens parcialmente superpostas.

Até essa data todos os artigos trabalhavam com imagens sem ruído, quando John Weng e Chitra Dorai construíram um método robusto de registro para múltiplas imagens de profundidade em presença de incertezas e ruído. Eles computavam os parâmetros de transformação com precisão utilizando o que eles denominaram “estimador de variância mínima” (Dorai et al., 1994) melhorando, com essa técnica, o critério de distância não ponderada no registro.

Outros estudos utilizando outras características dentro da imagem foram feitos por Kari Pulli, que abordou o problema da digitalização de cores com a geometria dos objetos (Pulli, 1997), exibindo uma imagem realista dos objetos digitalizados a partir de pontos de vista arbitrários.

2.3 MÉTODOS DE DIGITALIZAÇÃO DE IMAGENS *3-D*

Existem vários tipos de digitalizadores, atualmente, com a tarefa de obter imagens de profundidade. Cada um deles utiliza diferentes tipos de sensores, mas a ideia é a mesma: obter imagens de profundidade de diversos tipos de objetos relacionando um sistema de coordenadas com a posição da câmera. A maioria dos sensores utilizados são sensores ópticos como câmeras ou digitalizadores a *laser*, mas existem outros tipos de sensores como sensores acústicos, magnéticos ou tácteis que também realizam a função de identificar essas distâncias (Jarvis, 1983).

Existe uma classificação dos tipos de digitalizadores encontrados no mercado (Varaday, et al., 1997). Com ajuda dessa classificação, o pesquisador pode ter uma ideia de qual digitalizador usar dependendo das condições necessárias do seu sistema de estudo. A

principal classificação baseia-se fundamentalmente no tipo de sensor utilizado, que se divide em digitalização por contato ou sem contato. Na digitalização por contato é utilizado um braço de medição, como mostra a Figura 2.3. Esse tipo de digitalização pode ser não destrutiva quando o objeto digitalizado é conservado e pode também ser destrutiva quando o objeto digitalizado é destruído (geralmente utilizada quando se precisam digitalizar as partes internas de um objeto).

Os métodos de contato chamados "*Contact 3-D Scanners*" que fazem a reconstrução ou reconhecimento de objetos têm dois problemas específicos: o primeiro é que a reconstrução ou o reconhecimento podem alterar o objeto sobre o qual estão trabalhando, e o segundo é a perda de informação do objeto devido à existência de regiões de difícil acesso (Lerchet al., 2007).

Figura 2.3 Digitalização por contato.

A outra classificação de digitalizadores é a digitalização sem contato (Boehnen e Flynn, 2005). Nessa classificação, podem-se encontrar digitalizadores que utilizam sensores ópticos, acústicos e magnéticos. Nos sensores ópticos encontram-se digitalizadores a laser, dispositivos que realizam uma varredura a laser para fazer a medição e a digitalização remota de alta precisão (ver Figura 2.4).

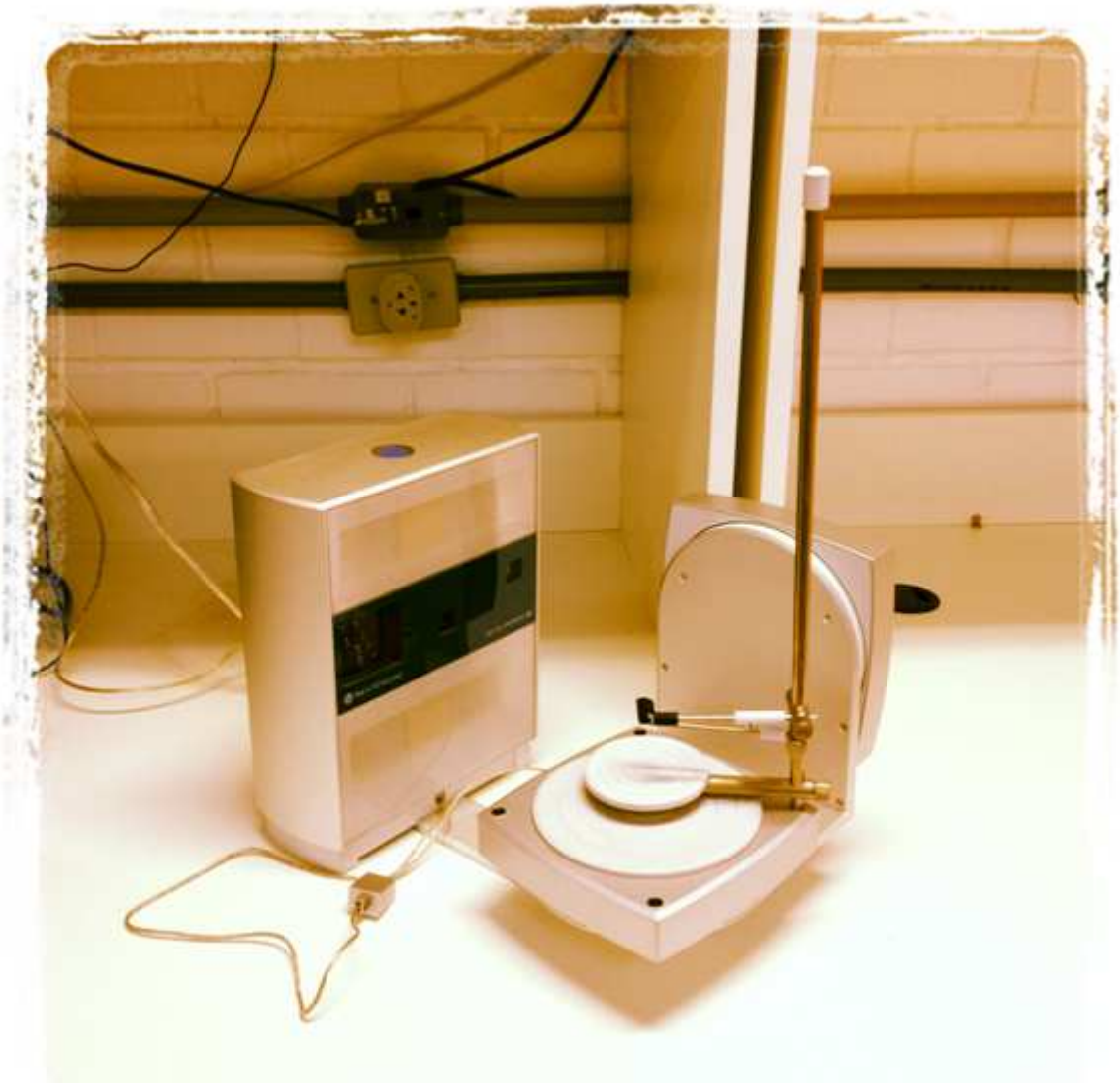


Figura 2.4 Dispositivo de varredura a laser. Laboratório de prototipagem rápida e engenharia reversa (Universidade de Brasília).

Para que uma varredura a laser seja feita, é preciso controlar a iluminação do sistema, já que depende de características ópticas como brilho e refletância.

2.4 APLICAÇÕES DO REGISTRO E MODELAMENTO DE OBJETOS

O desenvolvimento de técnicas de reconstrução, como o registro e modelamento de objetos, oferece uma expansão tecnológica em diversas áreas da engenharia. Embora seja

uma área muito nova, está oferecendo mudanças em nosso "*modus vivendi*", já que nossa vida está tendo uma inserção de aplicações inovadoras, como o "*Google Goggles*®" (criada pela *Google*®). Este programa permite reconhecer qualquer objeto ao registrar imagens mediante fotos tiradas com um celular e retornar resultados e informações relacionadas com ele, mudando o mundo dos viajantes (ver Figura 2.5), já que oferece muita informação da cultura, do estilo de vida e da arte de qualquer região, sem precisar de um guia humano.

Figura 2.5 Exemplo de como funciona o "*Google Goggles*".

O processo da modelagem de objetos não é um campo que pertence totalmente ao campo da visão computacional, mas muitas outras áreas também são inclusas, por exemplo, a matemática, a estatística, a inteligência artificial, a computação gráfica, entre outras. A visão computacional tem como objetivo recriar, interpretar e recuperar informações das cenas do mundo real a partir de imagens. Esse objetivo atualmente é muito útil dentro de diversas áreas como:

- Medicina: o uso de modelos digitais pode oferecer uma nova visão aos profissionais da saúde ao representar virtualmente órgãos corporais para sua manipulação sem a destruição

destes, além de proporcionar ideias na invenção de novas técnicas cirúrgicas e servir como apoio em técnicas antropométricas (Figura 2.6). Alguns exemplos de modelos digitais são as imagens de ecografia, imagens de tomografias axiais computadorizadas, ultrassom e ressonâncias magnéticas (Sandu e Topala, 2007).

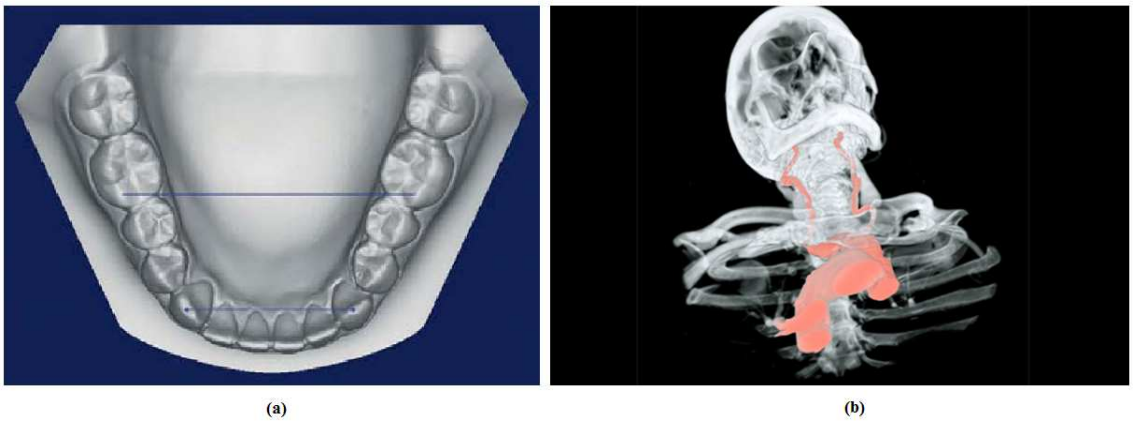


Figura 2.6 (a) Aplicação de técnicas antropométricas na odontologia. (b) Modelagem de órgãos humanos com finalidades educativas.

- Videojogos: nos videojogos mais avançados estão sendo implementadas técnicas de visão computacional para reconstruir partes do corpo de atores ou jogadores profissionais reais. Esse é o caso dos jogos de futebol, onde se realizam digitalizações dos rostos de jogadores famosos (Figura 2.7), além do corpo e os movimentos típicos que eles realizam (Malhotra e Gupta, 2011).

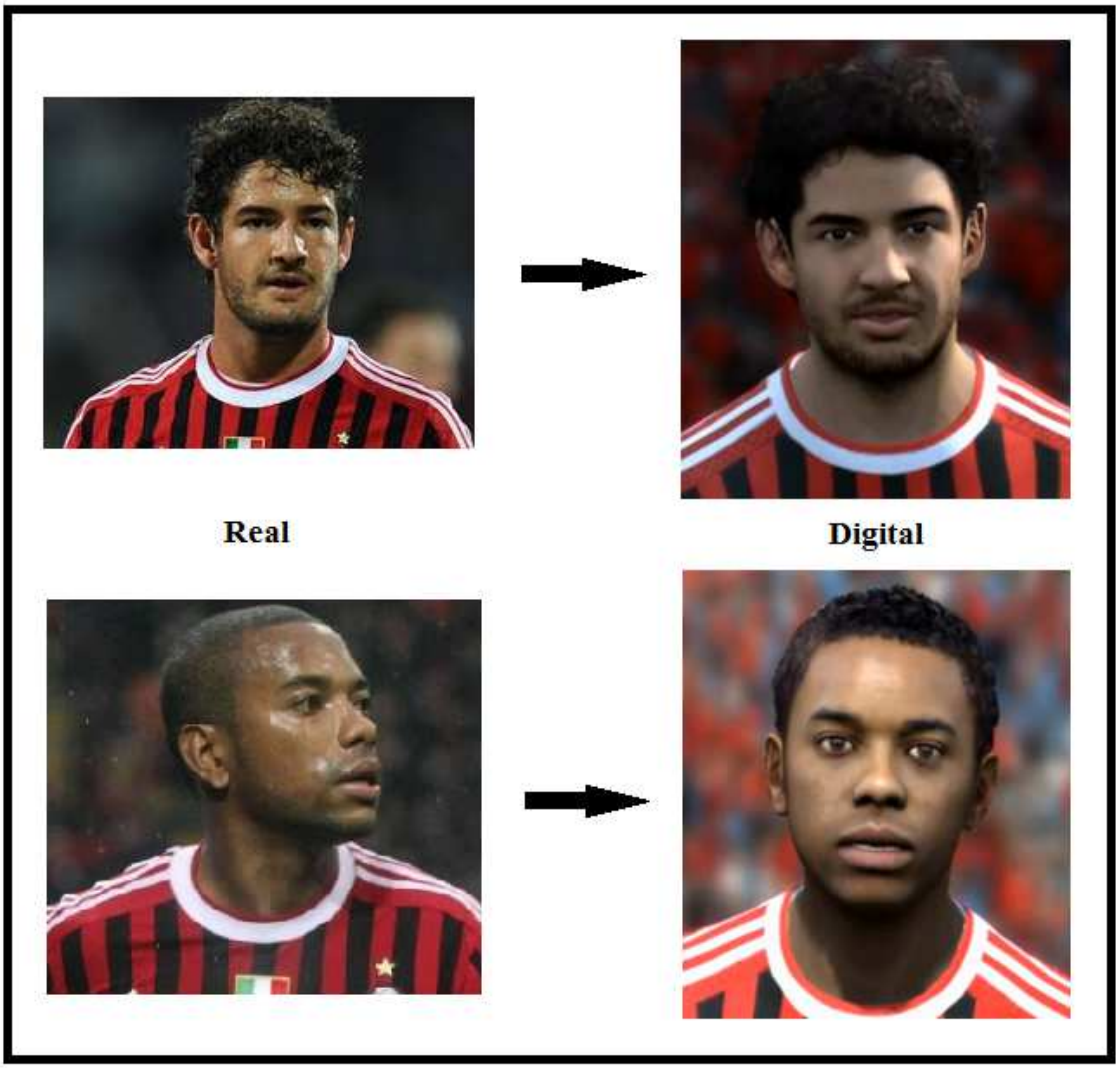


Figura 2.7 Modelamento de rostos para videojogos.

- A navegação robótica é o termo que representa um conjunto de ações que permitem guiar a trajetória de alguns robôs que têm a capacidade de se movimentar e se deslocar inclusive sobre ambientes desconhecidos. Por esse motivo muitos pesquisadores têm investido muitos esforços para oferecer sistemas que permitam orientar robôs dentro de diferentes tipos de ambientes (Gomes, 2001). A ideia fundamental é disponibilizar nos robôs sistemas que concebam a capacidade de navegar por diferentes ambientes de maneira automatizada e sem a ajuda do homem. O registro e modelamento de objetos ajudam à navegação robótica a resolver problemas de reconhecimento de trajetórias e soluções de obstáculos em sistemas desconhecidos (ver Figura 2.8).

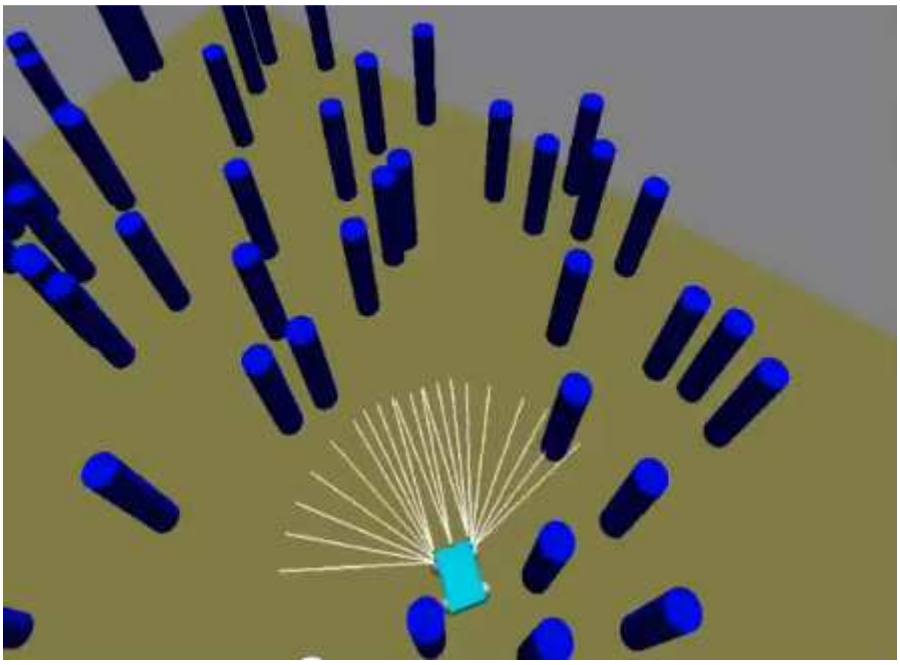


Figura 2.8 Reconstrução do ambiente para reconhecimento de trajetórias e soluções de obstáculos.

A navegação robótica pode ser feita utilizando sensores óticos baseados em laser como digitalizadores ou também câmeras fotométricas como as *CCD*. A grande vantagem dos digitalizadores de luz estruturada é que proporcionam uma melhor resolução espacial que as câmeras fotométricas, mas não proporcionam informação de outro tipo de características como a cor dos objetos digitalizados. É por esse motivo que a fusão de dispositivos como digitalizadores de luz estruturada e câmeras fotométricas oferecem um entendimento cognitivo do ambiente de estudo mais aproximado.

Nos últimos anos tem sido feitos também outros estudos utilizando modelos *3-D* obtidos por digitalizadores a laser com outros sistemas de navegação (Hähnel, et al., 2003), como o registro de imagens utilizando robôs moveis, Figura 2.9 (a), para adquirir modelos *3-D* de objetos estacionários, 2.9 (b), com a ajuda de mapas do ambiente, 2.9 (c), que ajudam na localização do robô dentro do ambiente de estudo.

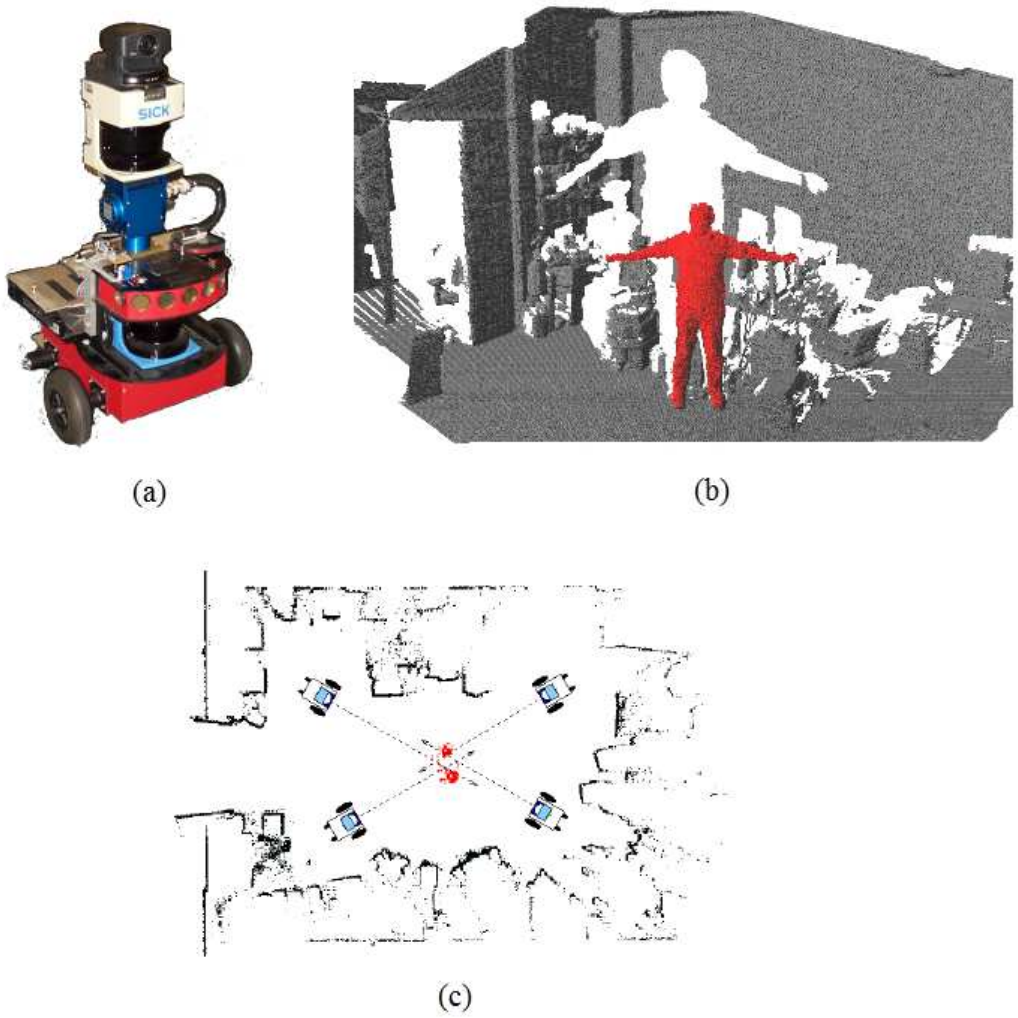


Figura 2.9 (a) Robôs moveis. (b) Aquisição de modelos 3-D. (c) Mapas do ambiente.

De forma geral, as técnicas de visão computacional de registro e modelamento de objetos podem ser muito úteis em uma grande quantidade de domínios dentro de vários campos da indústria, sendo usados cada vez mais e melhorando com o tempo com os avanços tecnológicos.

3. DEFINIÇÕES E CONCEITOS DO REGISTRO DE IMAGENS

3.1 TIPOS DE IMAGENS

3.1.1 Imagens de intensidade

Esse tipo de imagem mede a quantidade de luz que incide sobre um dispositivo fotossensível (Trucco e Verri, 1998). Qualquer imagem digital do tipo imagem de intensidade é uma matriz 2-D de números (ver Figura 3.1).

117	125	133	127	130	130	133	121	116	115	100	91	93	94	99	103	112	105	109	106
134	133	138	138	132	134	130	133	128	123	121	113	106	102	99	106	113	109	109	113
146	147	138	140	125	134	124	115	102	96	93	94	99	96	99	100	103	110	109	110
144	141	136	130	120	108	88	74	53	37	31	37	35	39	53	79	93	100	109	116
139	136	129	119	102	85	58	31	41	77	51	53	53	33	37	41	69	94	105	108
132	127	117	102	87	57	49	77	42	28	17	15	13	13	17	41	53	69	88	100
124	120	108	94	72	74	72	31	35	31	15	13	15	11	15	13	46	75	83	96
125	115	102	93	88	82	42	79	113	41	19	100	82	11	11	17	31	91	99	100
124	116	109	99	91	113	99	140	144	57	20	20	15	11	15	17	63	87	119	124
136	133	133	135	138	133	132	144	150	120	24	17	15	15	17	20	115	113	88	150
158	157	157	154	149	145	133	127	146	150	116	35	20	19	28	105	124	128	141	171
155	154	156	155	146	155	154	154	147	139	148	150	138	120	128	129	130	151	156	165
150	151	154	162	166	167	169	174	172	167	177	166	164	140	134	120	121	120	127	172
145	149	151	157	165	169	173	179	176	166	166	157	145	136	129	124	120	136	163	168
144	148	153	160	159	158	165	172	165	169	157	151	149	141	130	140	151	162	169	167
144	141	147	155	154	149	156	151	157	157	151	144	147	147	149	159	158	159	166	165
139	140	140	150	153	151	150	146	140	139	138	140	145	151	149	156	156	162	162	161
136	134	138	146	156	164	153	146	145	136	139	139	140	141	149	157	159	161	169	166
136	133	136	135	144	159	168	159	151	142	141	145	139	146	153	156	164	167	172	168
133	129	140	142	146	159	167	165	154	151	146	141	147	154	156	160	161	157	153	154

(a)

(b)

Figura 3.1 (a) Exemplo de uma Imagem de intensidade. (b) Matriz de dados da imagem de intensidade, (Trucco e Verri, 1998).

Esse tipo de informação é codificada dependendo do sensor usado para registrar as imagens.

As imagens de intensidade têm diversas limitações, já que não é possível ter uma medida das distâncias entre os objetos e componentes que fazem parte da cena.

3.1.2 Imagens de profundidade

Esse tipo de imagem estima diretamente a estrutura *3-D* da cena vista através de uma variedade de técnicas. Cada *pixel* expressa a distância entre um sistema de coordenadas de referência em um ponto visível na cena. Portanto, uma imagem de profundidade reproduz uma estrutura *3-D* de uma cena, sendo que um quadro é cada uma das imagens fixas de um produto audiovisual (Trucco e Verri, 1998).

3.1.3 Representação das imagens de profundidade

As imagens de profundidade podem ser representadas basicamente de duas formas. Um tipo de representação pode ser feita com uma lista de coordenadas x, y e z (*3-D*) em um quadro de referência, sem especificar a ordem dos dados. Este tipo de representação é chamada “nuvem de pontos”. O outro tipo de representação é feita com uma matriz de valores de profundidade dos pontos ao longo das direções dos eixos da imagem, oferecendo uma informação espacial explícita. Esse tipo de representação é chamada “Forma r_{ij}”.

Neste trabalho foram utilizadas imagens de profundidade do tipo “nuvem de pontos” como mostra a Figura 3.2

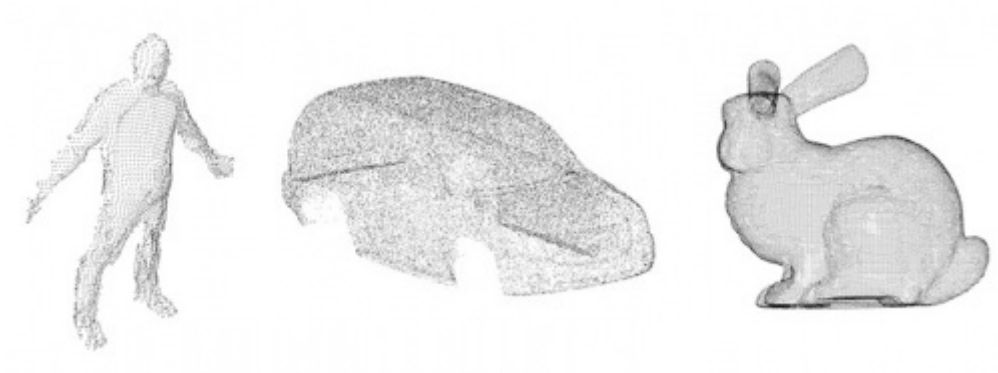


Figura 3.2 Renderização de imagens que representam nuvens de pontos.

3.2 CONCEITOS IMPORTANTES

3.2.1 *Dropout*

Os “*dropout*” (que vem do inglês “*Dropped Out*”) são regiões dentro de uma imagem nas quais o sinal se ausentou e não foi registrado. É comum observar “*dropouts*” quando ao tentar registrar imagens de superfícies de objetos se obtém superfícies com buracos. Um exemplo dessas imagens são as imagens obtidas com ultrassom (Tagare, 2006).

3.2.2 Transformação Rígida (Translação e Rotação)

Dentro de um sistema é imprescindível que todos os componentes trabalhem dentro do mesmo sistema de coordenadas. Esse fato permite uma livre manipulação dos objetos sem perder a localização no sistema. Quando os objetos têm diferentes coordenadas é preciso realizar uma transformação para que os sistemas de coordenadas independentes tenham uma relação direta que permita localizar os componentes desses objetos dentro de um só sistema de coordenadas. Em algumas técnicas de registro e reconstrução de objetos *3-D* são utilizadas mudanças de coordenadas dentro do sistema para alinhar os objetos que queiram ser digitalizados. Essas mudanças de coordenadas podem ser feitas alterando previamente os valores do sistema, ou podem ser feitas utilizando algoritmos iterativos.

No processo de registro é necessário representar diferentes conjuntos de dados (nuvens de pontos) em um único sistema de coordenadas, já que não necessariamente os conjuntos de dados são iguais, podendo ser de diferentes tipos, dependendo do registro que se precise. Para realizar um registro de imagens de um mesmo objeto é exigida uma transformação espacial de uma imagem objetivo a uma imagem de referência.

As imagens utilizadas neste trabalho para fazer o processo de registro foram nuvens de pontos obtidas por um digitalizador a laser. A imagem objetivo vai ser denominada de "nuvem de pontos Base" e a imagem de referência vai ser denominada de "nuvem de pontos Modelo". No processo, a posição da nuvem de pontos base foi modificada aplicando-se sobre ela uma transformação de corpo rígido, modificando suas coordenadas (no sistema de referência) para realizar o alinhamento entre as nuvens de pontos. Para modificar as coordenadas foram manipulados os parâmetros extrínsecos, aplicando uma translação e uma rotação sobre os eixos que determinavam a posição dessa nuvem (ver Figura 3.3). Esse processo é conhecido como transformação rígida.

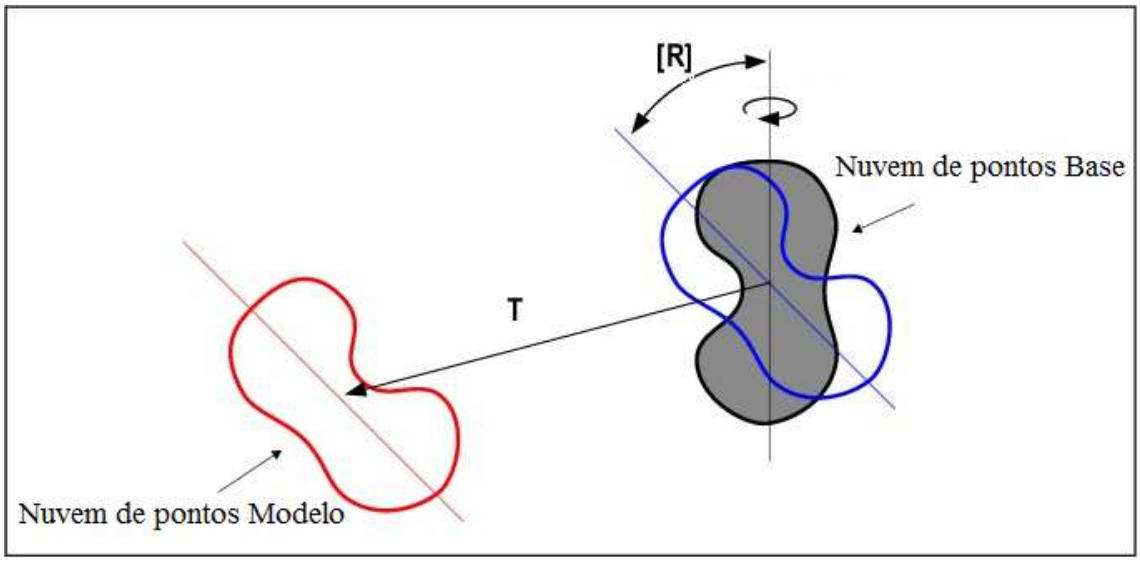


Figura 3.3 Exemplo de transformação rígida feita na nuvem de pontos Base. "[R]" é a rotação e "T" é a translação.

Uma transformação rígida sobre um vetor de espaço preserva a distância entre cada par de pontos. Esta pode ser no plano R^2, no espaço R^3, ou em um espaço real n-dimensional R^n. A transformação rígida é composta de rotações, translações, reflexões ou a combinação delas.

As transformações rígidas, além de serem muito utilizadas na área da robótica, também o são na área da visão computacional, especialmente para localizar objetos no espaço através da rotação e da translação.

3.2.3 *Kd tree*

Uma *kd tree* é uma estrutura de dados para armazenar um conjunto finito de pontos de um espaço k-dimensional (Bentley, 1980). É um tipo de "*binary tree*", conhecida amplamente nas ciências da computação, composta por "nós", "ramos" e "folhas". A sua estrutura é em forma de árvore, por isso seu nome em inglês "tree", da qual se desprendem nós, continua com as topologias próprias das árvores, herdando características. Em outras palavras, os seus dados estão dispostos de forma hierárquica.

Algumas outras características são as seguintes:

- O elemento principal de uma *tree* é chamado "raiz", este elemento possui ligação com outros elementos chamados "ramos" ou "filhos". Como exemplo, uma raiz dentro da figura 3.4 é formada pelos nós N1, N2, N4, N8 e N11.

- Cada nó tem até dois nós filhos.

- Os "ramos" estão conectados com outros elementos que também possuem outros "ramos".

- O elemento que não possui mais "ramos" é chamado como "folha". Como exemplo, as folhas dentro da Figura 3.4 são os nós N11, N5, N6, N10 e N12.

- A ordem de uma árvore é determinada pela quantidade máxima de ramos que possui em um elemento.

- Uma *kd tree* utiliza planos perpendiculares a um dos eixos do sistema de coordenadas. Além disso, todos os nós de uma *kd tree* armazenam um ponto. Desta forma cada plano deve passar através de um dos pontos da *kd tree*.

- Uma *kd tree* é uma estrutura de dados que organiza os pontos em um espaço euclidiano de k dimensões (Eskola, 2001).

- O custo computacional é proporcional ao número de níveis da *tree*.

- A complexidade computacional na criação de uma *kd tree* não é maior que O (*k N log2 N*). A complexidade computacional de construir uma *kd tree* a partir de "n" pontos é de O (nlogn).

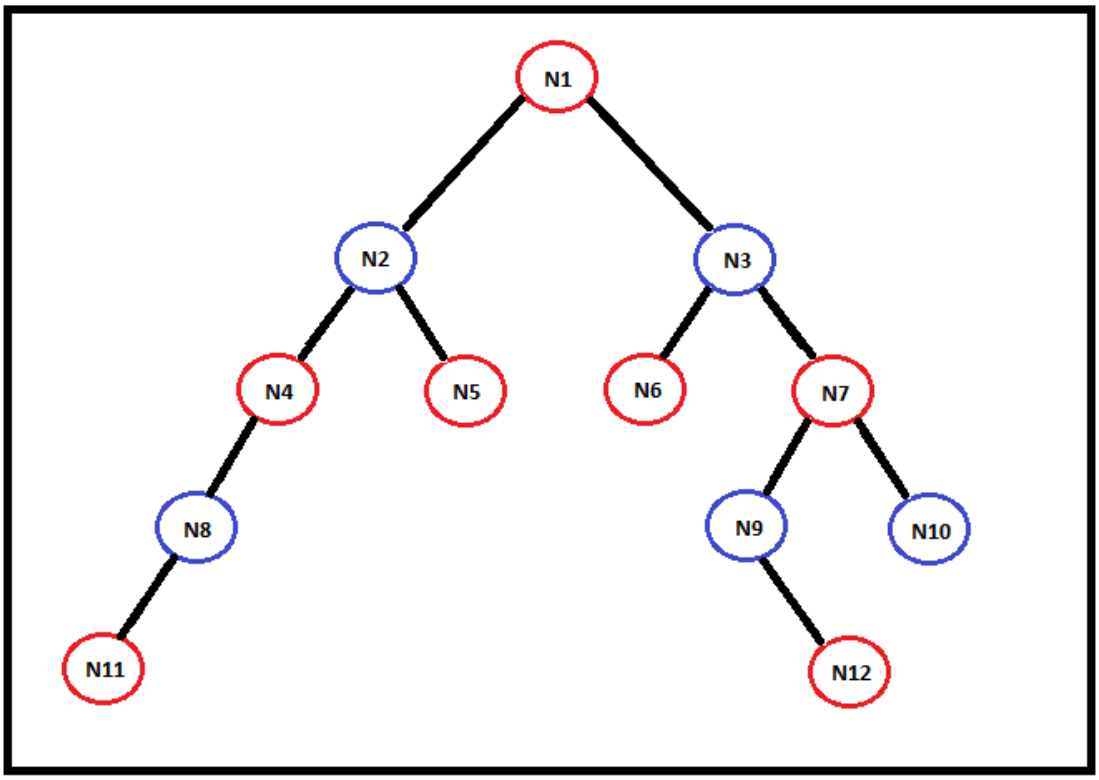


Figura 3.4 *Kd tree*, formada por doze nós.

Esta estrutura permite organizar todos os pontos de uma nuvem de pontos dentro de uma *kd tree* utilizando um processo chamado "*binary space partitioning*" ou "*BSP*" (ver Apêndice A). Primeiro é calculada a media das coordenadas de todos os pontos da nuvem e se escolhe o ponto que mais se aproxime nessa media. Esse ponto é considerado a "raiz" da *kd tree* e é ponto de partida para que sejam organizados todos os pontos dentro da *kd tree* mantendo a relação da proximidade entre eles (Bruce F. Naylor, 1993).

O nó N1 representa o ponto que mais se aproxima a média de todas as coordenadas dentro da nuvem de pontos (ver Figura 3.5).

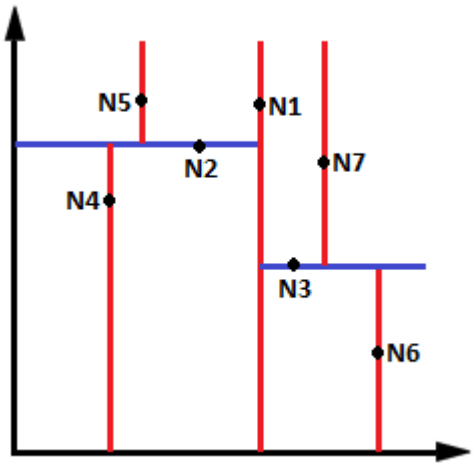


Figura 3.5 *BSP tree* formada com sete nós.

A Figura 3.5 constitui a *kd tree* da Figura 3.4 dos sete primeiros nós para pontos em dois dimensões.

3.3 O ALGORITMO *ICP*

O algoritmo *ICP* é um algoritmo iterativo que converge monotonicamente para o mínimo local mais próximo de uma função de custo para encontrar uma transformação rígida adequada entre duas imagens de profundidade (Chen e Medioni, 1992). Assim, esse método refina iterativamente uma transformação inicial tentando minimizar a função de custo medindo a similaridade entre duas nuvens de pontos. As iterações terminam quando um critério de parada é atingido. É um método computacionalmente eficiente no registro de objetos *3-D*, não só para figuras geométricas perfeitas se não também para objetos com curvas e superfícies livres (Besl e McKay, 1992).

Szymon Rusiskiewicz e Marc Levoy (2001) identificaram seis etapas dentro do algoritmo, além de enumerar e classificar de forma detalhada diferentes variantes do algoritmo *ICP*: “Seleção” (*Selection*), “Correspondência” (*Matching*), “Pesagem” (*Weighting*), “Rejeição” (*Rejecting*), “Erro métrico” (*Metric error*) e “Minimização” (*Minimization*).

3.3.1 Etapas do algoritmo *ICP*

3.3.1.1 Seleção

A etapa de seleção considera somente alguns dos dados (pontos do modelo) para aplicar o algoritmo *ICP*. Isso poderia ser benéfico, já que na maioria dos casos não se precisa utilizar todos os dados para conseguir uma boa transformação rígida, melhorando o custo computacional. Nessa etapa podem ser filtrados pontos que podem ser considerados como "*outliers*" (ver Apêndice B) com algum limiar como a distância máxima entre pontos correspondentes ou filtrá-los manualmente antes de colocar as imagens dentro do algoritmo. A intenção é fazer uma amostragem diferente em cada iteração do algoritmo, para evitar qualquer tendência para valores extremos.

Quando se tem informação sobre a cor dos objetos, normais tangentes e curvaturas, o armazenamento dessas características representadas em alguma distribuição pré-definida pode ser a melhor estratégia para descartar informação redundante presente dentro das nuvens de pontos.

Um registro inicial pode ser feito antes dessa etapa. Dependendo do grau de conhecimento que se tenha do ambiente do objeto, pode-se aproveitar a localização do objeto, para evitar os custos dos processos do algoritmo *ICP* ao selecionar os pontos que não fossem de interesse para fazer a minimização.

Na literatura existem varias técnicas para realizar a seleção de pontos, como usar todos os pontos disponíveis (Besl e Mckay, 1992), fazer uma subamostragem dos pontos disponíveis (Turk e Levoy, 1994), fazer uma amostragem aleatória (Masuda, et al., 1996), selecionar os pontos com alto grau de intensidade (Weik, 1997) ou selecionar pontos de ambas as malhas (Godin, 1994). Existem dois tipos de técnicas especiais que podem se utilizar para realizar a seleção:

- Um registro baseado na forma dos objetos, aplicado por Simon (Simon, 96), desenvolvendo vários algoritmos especialmente adaptados para lidar com dados com ruído quando as áreas correspondentes são somente um subconjunto da malha de entrada, para selecionar os pontos em uma das malhas de entrada que tiveram o melhor potencial para restringir todas as transformações quando a outra malha seja alinhada com ela. Esses algoritmos foram desenvolvidos para casos quando só um número muito pequeno de

pontos é requerido para o alinhamento. Como resultado eles também são muito caros para ser usados quando um grande número de pontos está para ser selecionado para a minimização.

- Um registro utilizando uma técnica chamada "*Normal-Space Sampling*", proposto por Szymon Rusiskiewicz e Marc Levoy (2001), que visa restringir o deslizamento translacional das malhas de entrada. O algoritmo pode ser visto como tratando de equalizar os autovalores dos autovetores de C que correspondem às translações.

3.3.1.2 Correspondência

A correspondência visa encontrar os pontos correspondentes entre os pontos da nuvem de pontos base com os pontos da nuvem de pontos modelo.

Encontrar os pontos correspondentes é usualmente a etapa mais cara computacionalmente do algoritmo *ICP*. Um dos métodos de busca chamado "Força bruta" calcula as distâncias entre um ponto de uma imagem com cada um dos pontos da outra imagem, escolhendo o par de pontos com a menor distância. Esse método funciona tanto para imagens de profundidade como para imagens de intensidade (Enqvist et al., 2011). Outros métodos podem ser implementados melhorando a velocidade para encontrar os pontos correspondentes. O método "*kd tree*" provê tempos de busca menores em alguns pré-processos comparado com o método de busca "força bruta" (Birn et al., 2010). O "*kd tree*" divide os pontos da imagem dependendo da sua localização, economizando cálculos de distâncias entre os supostos pontos correspondentes.

A correspondência pode ser feita especialmente de duas formas:

-Ponto a ponto (ver Figura 3.6), na qual a ideia é procurar os pontos correspondentes buscando a menor distância entre um ponto da nuvem de pontos base e todos os pontos da nuvem de pontos modelo e, assim, resolver um problema de otimização chamado "*Nearest Neighbor*" (Hwang, et al., 2012) com diferentes técnicas como a "triangulação de Delaunay" ou como a "*kd tree*".

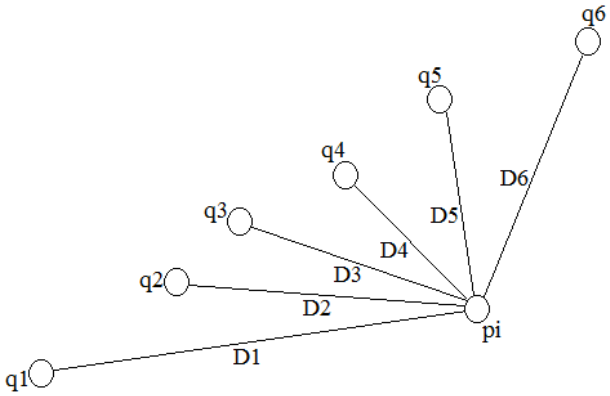


Figura 3.6 Correspondência ponto a ponto.

Essa implementação é fácil de construir, mas nem sempre os pontos correspondentes entre a nuvem de pontos base e a nuvem de pontos modelo são os mais próximos entre si, convertendo esse problema em uma heurística (ver Figura 3.7).

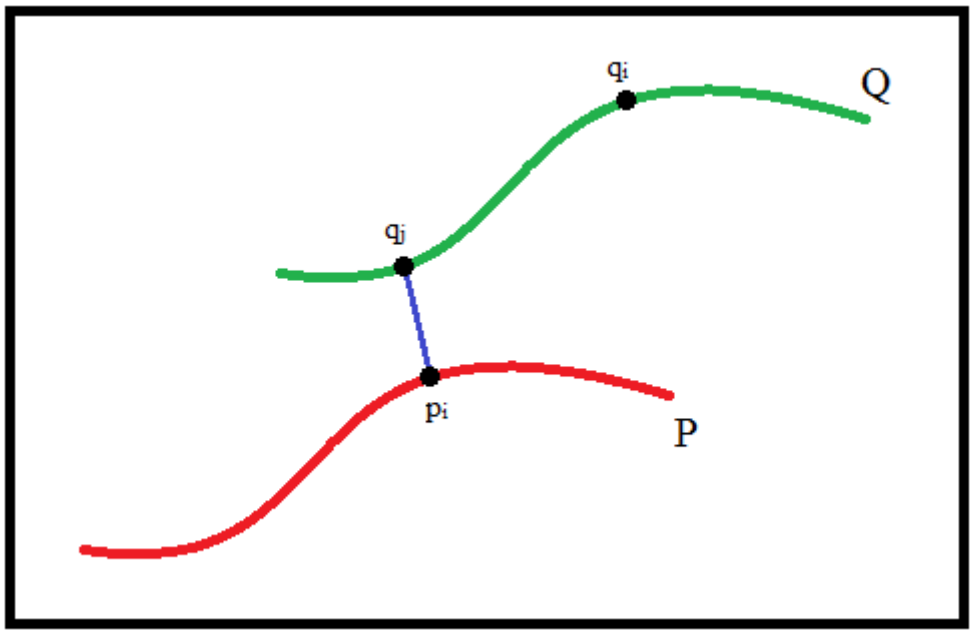


Figura 3.7 Falsos pontos correspondentes entre duas imagens que representam a mesma superfície.

Na Figura 3.7 o verdadeiro ponto correspondente da imagem Q para o ponto p_i da imagem P é o ponto q_i e não o ponto q_j. Nesses casos, os falsos pontos correspondentes devem ser

descartados ou se realizar um adequado alinhamento inicial entre as duas imagens (ver Figura 3.8), para que assim, os pontos correspondentes sejam os pontos mais próximos entre as duas imagens.

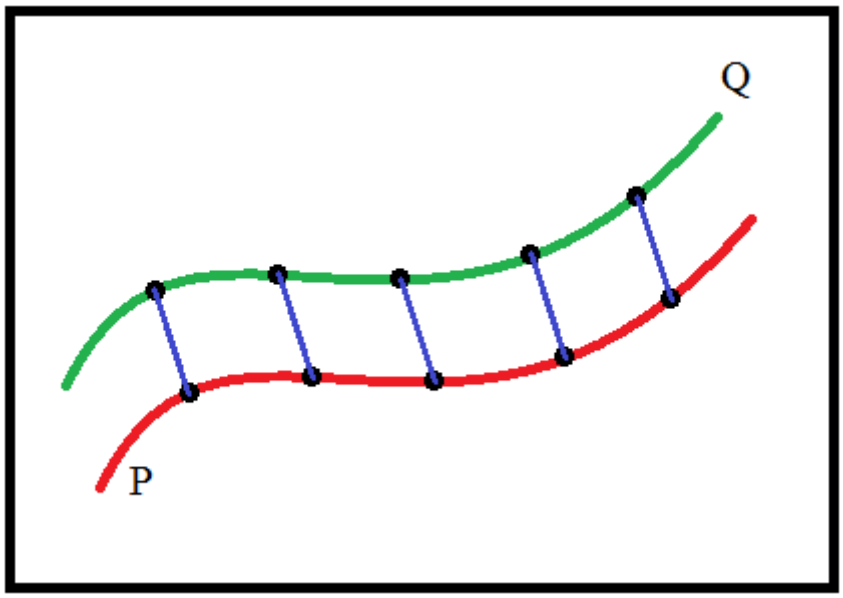


Figura 3.8 Correspondência adequada entre duas imagens correspondentes à mesma região do objeto produto de um bom alinhamento inicial.

- Ponto a plano, utiliza uma função de custo que relaciona a distância entre um ponto da imagem base e o plano da imagem modelo na direção da normal ao plano no ponto da imagem modelo. Alguns dos trabalhos que implementam correspondência ponto ao plano são os trabalhos de Chen e Medioni, (1992), Rusinkiewicz e Levoy (2001) e Bae e Lichti, (2008). O meio de correspondência dos algoritmos baseados em ponto a plano é estimar os parâmetros de transformação relativa que minimizem a soma das distâncias quadráticas entre os pontos e suas correspondentes superfícies (Chen e Medioni, 1992), como visto na Figura 3.9. Essa correspondência utiliza uma técnica chamada "*Normal Shooting*", que encontra os pontos correspondentes calculando a intersecção entre as normais dos pontos, da nuvem de pontos base, até a linha que representa a superfície do objeto na nuvem de pontos modelo. Esse método pode ser o melhor quando os dados têm muito ruído, mas requer também um bom alinhamento inicial (Rusinkiewicz e Levoy, 2001).

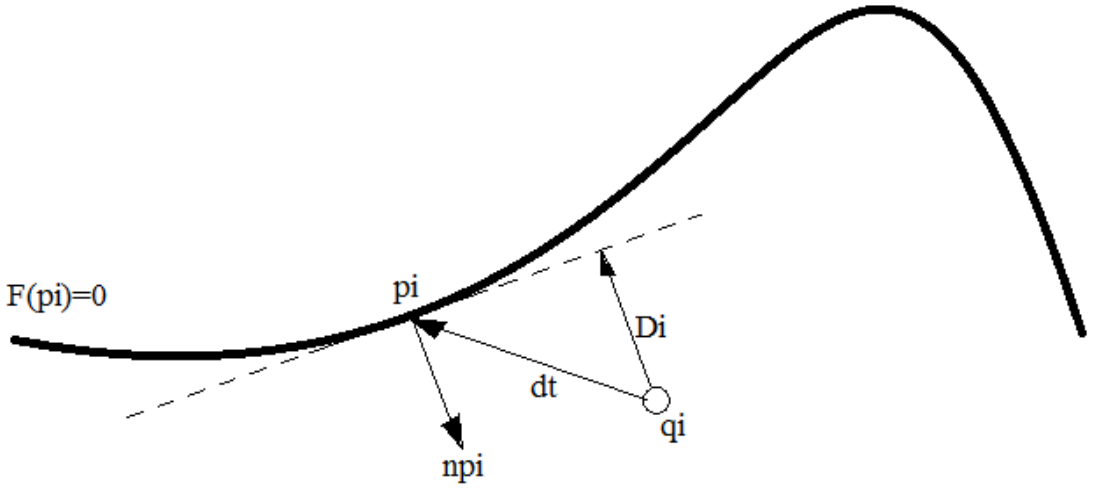


Figura 3.9 Correspondência ponto ao plano, onde "$F(p_i)$" representa a função do plano, "p_i" representa o ponto da nuvem de pontos modelo e "q_i" representa o ponto da nuvem de pontos base. A direção da distância "D_i" é paralela à direção da normal "n_{pi}" do ponto p_i.

A correspondência ponto ao plano geralmente é mais resistente ao ruído (ver figura 3.10), já que os pontos correspondentes são identificados calculando a menor distância entre um ponto de uma imagem e a superfície da outra imagem utilizando "*Normal Shooting*".

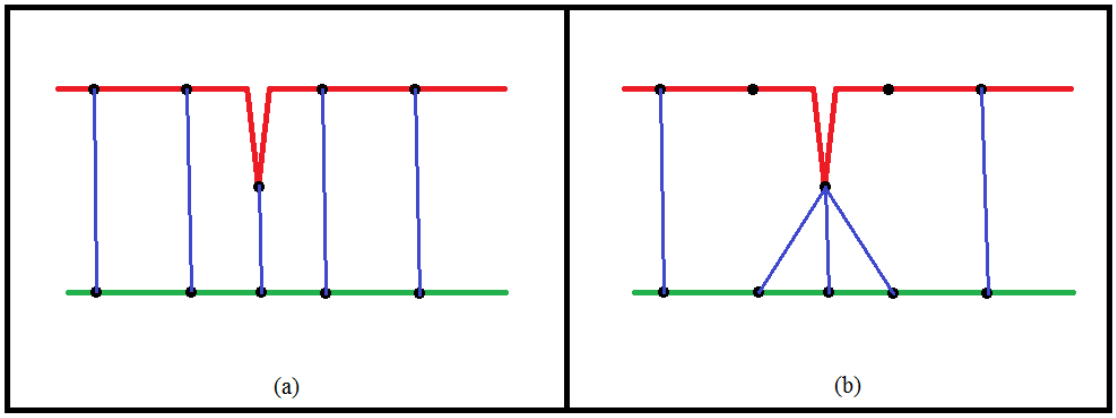


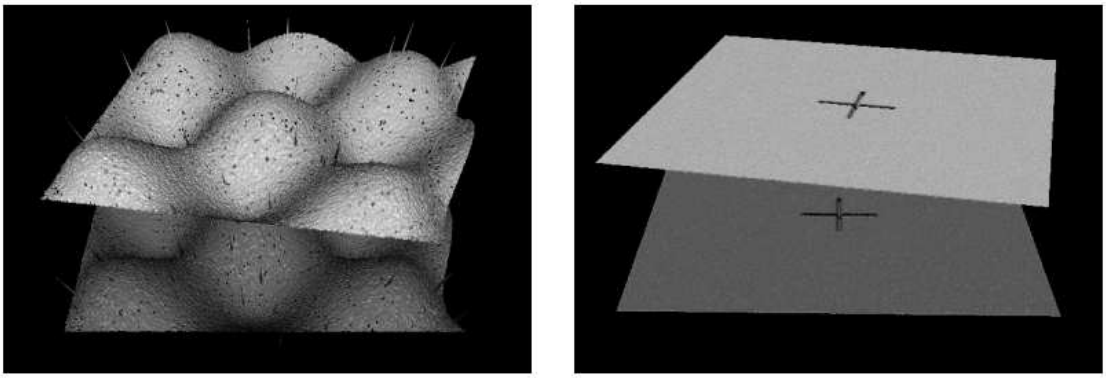
Figura 3.10 Diferença de correspondências de duas imagens. (a) Ponto ao plano. (b) Ponto a ponto.

3.3.1.3 Pesagem

A pesagem em um algoritmo tem como objetivo fazer ajustes em algumas variáveis para ajudar o algoritmo a convergir. Este tipo de ajuste é feito no algoritmo *ICP* sobre os pontos correspondentes, eles podem ser pesados de forma diferente dependendo da sua compatibilidade. A pesagem é feita multiplicando cada termo pelo erro métrico com um fator especifico *w*, escolhido com base em características do sistema como a distância, a cor, a curvatura ou a direção normal tangente (Rusinkiewicz e Levoy 2001).

A pesagem pode ser feita de quatro formas possíveis. Pode ser um peso constante (comumente chamada pesagem “Constante”), pode-se pesar com base no ruído esperado produzido pelo digitalizador na incerteza do erro métrico (comumente chamada pesagem por Incerteza), pode-se colocar os pesos com base na compatibilidade das normais dos pontos (comumente chamada pesagem por “Compatibilidade de normais”), ou pode-se atribuir pesos baixos aos pares com maiores distâncias ponto a ponto (comumente chamada pesagem “linear com distância”) como foi feito por Guy Godin (1994).

Szymon Rusinkiewicz e Marc Levoy (2001) consideram três tipos de cena dependendo do grau de dificuldade que tenha o algoritmo *ICP* para convergir para um bom resultado analisando o erro *RMS*. A cena que permite uma melhor convergência é chamada de “Onda” (*Wave*),que tem uma forma geométrica de superfície áspera alisada com adição de “*Outliers*”e ruído Gaussiano (Figura 3.11a). Outro tipo de cena é o “Plano de incisão” (*Incised Plane*), que consiste de dois planos com ruído Gaussiano e duas ranhuras em forma de um “X” (Ver Figura 3.11b), sendo esta cena mais complicada para a convergência do algoritmo ICP, já que a maioria dos algoritmos variantes não convergem para um alinhamento correto.



(a) Onda (b) Plano de incisão

Figura 3.11 Tipos de imagem descritos por Szymon Rusisnkiewicz e Marc Levoy (2001). (a) tipo “Onda”, (b) tipo “Plano de incisão”.

Quando se adiciona um ruído extra nas cenas “Onda” e “Plano de incisão”, é possível observar que os métodos de “incerteza” e “compatibilidade de normais” têm um melhor desempenho medido pelo erro *RMS*, convergindo mais rápido que as pesagens “constante” e “linear com distância”.

3.3.1.4 Rejeição

Nesta etapa se excluem os pontos correspondentes com base em um critério como, por exemplo, os pontos correspondentes que excedam uma distância mínima (considerada pelo observador conhecendo o ambiente do objeto) serão rejeitados para descartar pares falsos ou para eliminar principalmente os “outliers” que podem ter um grande efeito no desempenho na etapa da minimização utilizando o método dos mínimos quadrados (ver apêndice C). A rejeição de pontos depende das imagens obtidas, já que na criação de nuvens de pontos podem-se introduzir medições errôneas causadas pelo sensor (como dificuldade em acessar em certas regiões do objeto) ou pela falta de condições do ambiente (como a iluminação).

A rejeição de pontos é feita geralmente utilizando uma função de minimização ou encontrando os pontos a rejeitar segundo as características dos pares correspondentes. Na literatura se encontram vários métodos como:

a) Rejeitar só uma pequena porcentagem dos pontos correspondentes baseados em uma métrica usualmente uma distância ponto a ponto (Pulli, 1999);

(b) Rejeitar os pares correspondentes cuja distância ponto a ponto seja maior que alguns múltiplos da distância do desvio padrão (Masuda, et al., 1996);

(c) Os pares que não sejam consistentes com os pares vizinhos (Dorai, et al., 1998);

(d) Rejeitar os pares que tenham pontos sobre os limites da malha (Turk e Levoy, 1994).

3.3.1.5 Erro Métrico

O erro métrico define a função de custo que vai ser minimizada em cada iteração do algoritmo.

Na literatura têm sido usadas métricas de erro como:

(a) Soma das distâncias quadradas entre pontos correspondentes (Arun, et al., 1987) baseados em "*Singular Value Decomposition*" (ver apêndice D);

(b) *Quaternions* (Horn 1987);

(c) Matrizes ortonormais (Horn, et al., 1988);

(d) Métrica ponto a ponto, considerando as distâncias entre os pontos e a diferenças nas cores (Johnson e Kang1997);

(e) A soma das distâncias quadradas de cada ponto de origem para o plano que contém o ponto de destino (Chen e Medioni 1991).

3.3.1.6 Minimização

No algoritmo *ICP* é feita uma minimização sobre uma função de custo. Esta função de custo é aplicada iterativamente com os dados que estimam os parâmetros de corpo rígido (matriz de rotação e vetor de translação) com o objetivo de encontrar a melhor transformação.

Na literatura existem várias soluções para o problema da minimização da função de custo:

(a) Gerar repetidamente um conjunto de pontos correspondentes usando a transformação

atual e encontrando uma nova transformação que minimize o erro métrico (Chen e Medioni, 1991).

(b) Adicionando uma extrapolação no espaço de transformação para acelerar a convergência (Besl e Mckay, 1992);

(c) Desenvolver uma minimização iterativa começando com várias perturbações nas condições iniciais escolhendo o melhor resultado (Simon, 1996);

(d) Selecionar o resultado mais ótimo usando "*least median of squares*" como métrica robusta ao desenvolver uma minimização iterativa usando vários subconjuntos de pontos selecionados randomicamente (Masuda et al., 1996);

(e) Busca estocástica para a melhor transformação usando "*Simulated annealing*" (Blais e Levine, 1995).

Comumente são utilizadas duas métricas: a minimização ponto a ponto e a minimização ponto a plano.

A minimização ponto a ponto (Johnson e Kang 1997) é a soma das distâncias quadradas dos pontos da nuvem de pontos base e a nuvem de pontos modelo, expressada como:

$$e = \sum_{i=1}^{N} \left\| \boldsymbol{R} p_i + \vec{T} - q_i \right\|^2 \qquad (3.1)$$

onde "e" é a métrica de erro (soma dos quadrados dos erros), "R" é a matriz de rotação, "T" é o vetor de translação, "p_i" são os pontos da nuvem de pontos base e "q_i" são os pontos da nuvem de pontos modelo.

A minimização ponto ao plano (Chen e Medioni 1992) é a soma das distâncias entre os pontos da nuvem de pontos base e os planos tangentes correspondentes da nuvem de pontos modelo, expressada como:

$$e = \Sigma_{i=1}^{N} \left\| \left(\boldsymbol{R} p_i + \vec{T} - q_i \right) \cdot \vec{n}_i \right\|^2 \quad (3.2)$$

Onde "n_i" representa as normais tangentes estimadas no i-esimo ponto da nuvem de pontos modelo.

Segundo Chen e Medioni (1992), na minimização ponto ao plano, tendo um conjunto de N pares de pontos correspondentes chamados pontos de controle, de duas imagens (nuvens de pontos), como mostra a Figura 3.8.

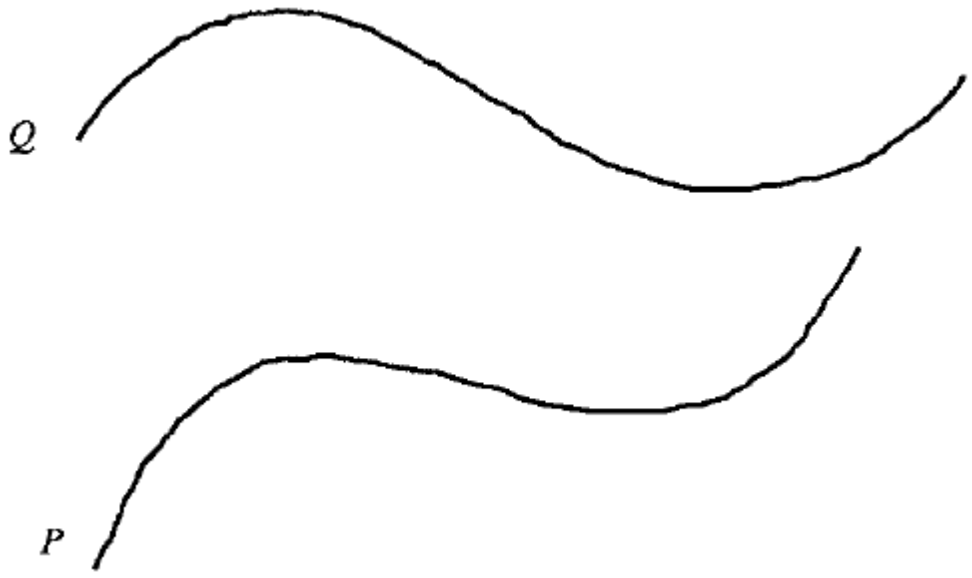


Figura 3.12 Nuvens de pontos, que mostram a distância entre P e Q antes de ser aplicada a transformação T.

Cada uma com pontos $p_i \in P$ e $q_i \in Q$, i=1...,N, respectivamente, a transformação necessária para realizar o alinhamento se encontra minimizando a seguinte expressão:

$$e = \sum_{i=1}^{N} \left\| \vec{T} p_i - q_i \right\|^2 \tag{3.3}$$

Minimizando as distâncias que existem entre os pontos correspondentes entre as duas imagens, obtem-se:

$$e = \sum_{i=1}^{N} \left\| \vec{T} p_i - q_j \right\|^2, com\ q_j = q \,|\, \min_{q \in Q} \left\| \vec{T} p_i - q_i \right\| \tag{3.4}$$

Tendo em conta que duas vistas de uma superfície estão registradas quando qualquer par de pontos (p_i , q_i) que representam o mesmo ponto na superfície das duas imagens coincidem dentro de um mesmo sistema de referência aplicando uma transformação T, assim:

$$\forall p_i \in P, \exists q_i \in Q \,|\, \left\| \vec{T} p_i - q_i \right\| = 0 \tag{3.5}$$

O algoritmo é melhorado ao implementar um método iterativo se é conhecida uma transformação inicial Tº que aproxime o registro entre P e Q. Nesse caso, em cada iteração k tem-se um valor prévio de T^{k-1} para encontrar q^k_j.Assim:

$$e^k = \sum_{i=1}^{N} \left\| T^k p_i - q_j^k \right\|^2, com\ q_j^k = q \,|\, \min_{q \in Q} \left\| T^{k-1} p_i - q_i \right\| \tag{3.6}$$

Com essa aproximação, é preciso realizar a minimização sobre a função para encontrar q^k_j. Usando a aproximação de ponto em vez do q^k_j definido na equação anterior, o problema fica mais fácil. Usando a distância entre as superfícies na direção normal à primeira superfície, como uma função de avaliação do registro (Potmesil, 1983):

$$e^k = \sum_{i=1}^{N} \left\| T^k p_i - q'^k_j \right\|^2, com\ q'^k_j = (T^{k-1}\ell_i) \cap Q \tag{3.7}$$

Onde

- $\ell_i = \{a | (p_i - a) \times n_{pi} = 0\}$, é a linha normal a P em p_i.

- n_{pi}, é a superfície normal de P em p_i.

- $(T\ell_i) \cap Q$, representa o ponto de interseção da linha ℓ_i (transformada depois por T) com a superfície Q.

O problema comum com a Equação 3.7 é que em cada iteração se trabalha com alguns pontos de controle que não necessariamente são verdadeiros pontos correspondentes, obtendo uma convergência lenta como consequência.

Então, outra maneira é aproximar Q usando seu plano tangente S_j em q_j da equação 3.4, obtendo:

$$e^k = \sum_{i=1}^{N} \left\| T^k p_i - q'_j \right\|^2, com\ q'_j = q | \min_{q \in S_j} \|T p_i - q_i\| \tag{3.8}$$

Onde S_j é o plano tangente de Q em q_j. Embora a posição de q_j seja desconhecida se uma transformação T^o é conhecida, é possível começar um processo iterativo como aproximação.

Assim, a distância entre o ponto e o plano pode ser expressa como uma função linear das coordenadas dos pontos, e o processo iterativo pode ser formulado da seguinte maneira:

$$e = \sum_{i=1}^{N} d_s^2 (T^k p_i, S_j^k) \tag{3.9}$$

com

$$S_j^k = \left\{ S | n_{qj}^k \cdot \left(q'^k_j - S \right) = 0 \right\}, q'^k_j = (T^{k-1} \ell_i) \cap Q$$

onde (ver Figura 3.9):

- *e*, é a função de custo.

- q^k_j, é o ponto de interseção entre Q com a linha $T^{k-1}L_i$.

- S^k_j, é o plano tangente ao ponto q'^k_j da nuvem de pontos Q.

- n^k_{qj}, é o vetor normal de q'^k_j na superfície Q.

- l_i, é a linha normal em p_i à superfície P.

- d_s, é a distância do ponto ao plano.

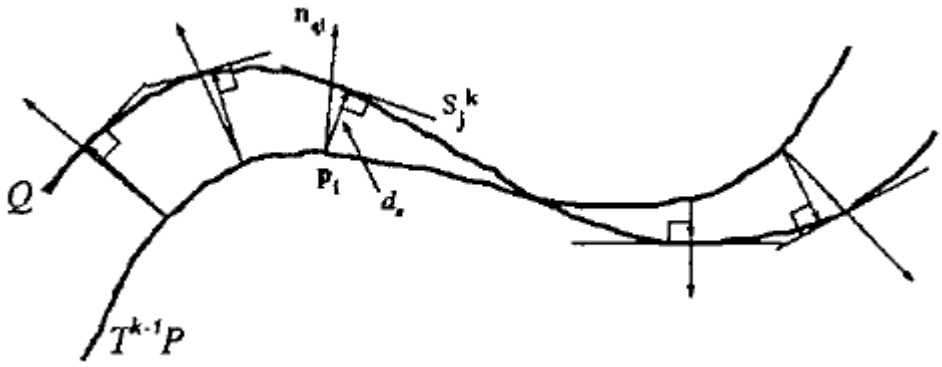


Figura 3.13 Nuvens de pontos, que mostram a distância entre P e Q depois de ser aplicada a transformação T^{k-1} sobre a imagem P. A função de custo e é formada pelas distâncias quadráticas d_s entre um ponto na superfície P a um plano tangente à superfície Q.

3.3.2 Descrição do *ICP* implementado

Neste projeto foram implementados dois tipos do algoritmo *ICP*. Um algoritmo *ICP* original que utiliza um método de busca chamado “força bruta” e outro algoritmo *ICP* melhorado com uma técnica de busca chamada *kd tree*.

Os dois algoritmos *ICP* implementados neste projeto utilizaram dois conjuntos de dados. Esses dois conjuntos de dados são duas nuvens de pontos que, neste trabalho, serão chamadas de nuvem de pontos base e nuvem de pontos modelo.

A grande diferença entre os dois algoritmos implementados está em que no algoritmo convencional (força bruta) mediram-se as distâncias entre a nuvem de pontos base e a nuvem de pontos modelo para cada um dos pontos, para posteriormente identificar a menor distância entre eles e assim encontrar os pontos correspondentes. De outra forma, no algoritmo melhorado (*kd tree*), primeiro se identificaram quais são os vizinhos mais próximos (utilizando o *BSP*) para posteriormente calcular as distâncias entre os pontos das duas imagens e assim identificar os pontos correspondentes diminuindo o custo computacional na etapa de correspondência.

A distância foi calculada para cada ponto da nuvem de pontos base com todos os pontos da nuvem de pontos modelo utilizando a Equação 3.10:

$$d(p,q) = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2 + (b_z - a_z)^2} \tag{3.10}$$

Tendo encontrado os pontos correspondentes entre as duas imagens, é feita a minimização ponto a ponto da função de custo representada na equação 3.1 (da seção 3.3.1.6) que corresponde à métrica de erro da seguinte forma (Arun, et al., 1987):

Primeiro são achados os centroides das nuvens de pontos base e modelo utilizando o número de pontos da nuvem de pontos base (n_b) e o número de pontos da nuvem de pontos modelo (n_m), além da somatória de todos os pontos de cada uma das nuvens como mostra a equação 3.11:

$$\bar{p} = \frac{1}{n_b}\sum p \quad , \quad \bar{q} = \frac{1}{n_m}\sum q \tag{3.11}$$

Como segundo passo se calcula o desvio p'_i e q'_i respectivamente para cada um dos pontos das nuvens, assim:

$$p_i' = p_i - \bar{p}, \quad q_i' = q_i - \bar{q} \tag{3.12}$$

Posteriormente utilizando a equação 3 e inserindo os termos encontrados nas equações 3.11 e 3.12, obtem-se:

$$e = \sum_{i=1}^{N} \left\| R(p_i' + \bar{p})\vec{T} - (q_i' + \bar{q}) \right\|^2 \tag{3.13}$$

E agrupando termos:

$$e = \sum_{i=1}^{N} \left\| Rp_i' - q_i'(R\bar{p} + \vec{T} - \bar{q}) \right\|^2 \tag{3.14}$$

Então, se o que se precisa é minimizar a função de custo, o vetor de translação deveria ajustar o centroide da nuvem de pontos base para o centroide da nuvem de pontos modelo. Essa afirmação seria representada na seguinte equação:

$$\vec{T} = \bar{q} - R\bar{p} \tag{3.15}$$

Substituindo a equação 3.15 na Equação 3.14 obtem-se:

$$e = \sum_{i=1}^{N} \| p_i' \|^2 - 2tr\left(R \sum_{i=1}^{N} p_i' \, q_i'^{T}\right) + \sum_{i=1}^{N} \| q_i' \|^2 \tag{3.16}$$

Onde para minimizar “e” podemos maximizar tr (RZ), que é o traço da matriz, onde Z é:

$$Z = \sum_{i=1}^{N} p_i' \, q_i'^{T} \tag{3.17}$$

Então, o traço de RZ pode ser expressado como:

$$tr(RZ) = \sum_{i=1}^{N} r_i \cdot c_i \leq \sum_{i=1}^{N} \|r_i\| \|c_i\| \tag{3.18}$$

Onde r_i e c_i são as colunas de Z e as linhas de R. Utilizando a desigualdade de Cauchy-Schwarz, pode-se dizer que:

$$tr(RZ) = tr(\sqrt{Z^T Z}) \tag{3.19}$$

Por ultimo aplicando o *SVD* (ver Apêndice A-4) sobre a Equação 3.19 obtem-se:

$$tr(VU^T U\Sigma V^T) = tr(V\Sigma V^{-1}) = tr\left(\sqrt{V\Sigma^{\mathrm{T}}\Sigma V^{-1}}\right) = tr(\sqrt{Z^T Z}) \tag{3.20}$$

Onde a rotação seria encontrada ao resolver:

$$R=VU^{T} \tag{3.21}$$

Obtendo desta forma o ultimo valor para calcular a métrica de erro “e”, junto com a matriz de rotação R e o vetor de translação T.

A pesagem e rejeição foram as duas únicas etapas que não foram feitas dentro do algoritmo, em principio porque se realizou um alinhamento inicial das imagens, alem de ser polidas na etapa inicial do seu processamento.

Em síntese, o algoritmo *ICP* implementado procura, iterativamente, determinar a correspondência entre os pares de pontos das nuvens de pontos modelo e de dados, estimando a função de custo que realize o melhor registro entre os dois conjuntos de dados.

3.4 ALGORITMOS IMPLEMENTADOS

3.4.1 Algoritmo *ICP* "força bruta"

OBJETIVO

Dadas duas superfícies "P" com pontos pi (i=1..N) e "Q" com pontos qi (i=1..M) de duas nuvens de pontos, determinar a melhor transformação "T" que minimize "e" utilizando a Equação 3.3 com o método dos mínimos quadrados iterativamente.

Os passos do algoritmo são os seguintes:

1- Calcular uma transformação inicial "Tº";

2- Para todos os pontos da imagem P calcular as distâncias até cada ponto da imagem Q;

3- Calcular a menor distância entre os pontos de P com os pontos da imagem Q para determinar os pontos correspondentes;

4- Minimizar a função de custo "e" calculando os centroides das imagens e aplicando a decomposição de valores singulares (*SVD*) sobre a métrica de erro:

$$e = \sum_{i=1}^{N} \left\| R p_i + \vec{T} - q_i \right\|^2$$

5- Calcular uma nova matriz de rotação R e um novo vetor de translação T e aplicá-los sobre a imagem P;

6- Repetir o processo desde o passo número 2 até que o número de iterações previsto termine;

7- Aplicar a transformação sobre a imagem P utilizando o ultima matriz de rotação e o ultimo vetor de translação calculados.

3.4.2 Algoritmo *ICP "kd Tree"*

OBJETIVO

Utilizando o algoritmo "*ICP*", modificar a etapa de "Correspondência" (passo número 2) encontrando os pontos correspondentes entre os dois conjuntos de pontos das nuvens de pontos de estudo. Dadas duas nuvens de pontos P e Q de duas nuvens de pontos:

1- Calcular o BSP *tree* da imagem "P".

2- Localizar os vizinhos mais próximos de "q" em "P" com os seguintes passos:

   a- Começando desde a raiz da *tree*, movimentar para baixo comparando as coordenadas a partir da atual dimensão até alcançar uma folha;

   b- Ao localizar uma folha marcar esse ponto como o melhor (temporalmente) e calcular a distância "d" entre o ponto "q" e o melhor nó na *tree*;

   c- Voltar, movimentando um nível para cima da *tree* e calcular a distância desde *q* até esse nó nesse nível. Se aquela distância é mais curta que a distância calculada anteriormente "d", definir esse nó como o melhor nó e modificar a distância *d* com o novo valor;

   d- Se, pelo contrario, a distância desde "q" ao melhor nó no mesmo plano dimensional é maior que a distância "d", então apague os nó dessa rama e continue movimentando para cima até alcançar a raiz;

   e- Se nenhum desses casos é satisfatório, procurar no outro ramo do nó da mesma maneira na *tree* inteira;

   f- Ao final, quando o nó superior (top) for alcançado e todos os ramos laterais necessários tenham sido avaliados e medidos, faça uma escolha da menor distância entre todas as distâncias dos pontos candidatos;

4. RESULTADOS

4.1 PROCESSAMENTO DAS IMAGENS

Existem diversos algoritmos para reconhecimento de objetos, alguns deles trabalham utilizando imagens em *2-D*, utilizados para reconhecer vários objetos como ferramentas, cédulas, até códigos de barras. Neste trabalho foram utilizadas nuvens de pontos que são a representação dos objetos dentro de um sistema de coordenadas. Cada ponto dentro do sistema de coordenadas corresponde a um ponto do objeto do sistema físico. A aquisição das nuvens de pontos pode ser feita por um “digitalizador de luz estruturada”, que é capaz de digitalizar o ambiente para ser estudado, ver Figura 4.1.

Figura 4.1 Conceito de escaneamento.

Os objetos utilizados neste projeto para obter as nuvens de pontos são mostrados na Figura 4.2:

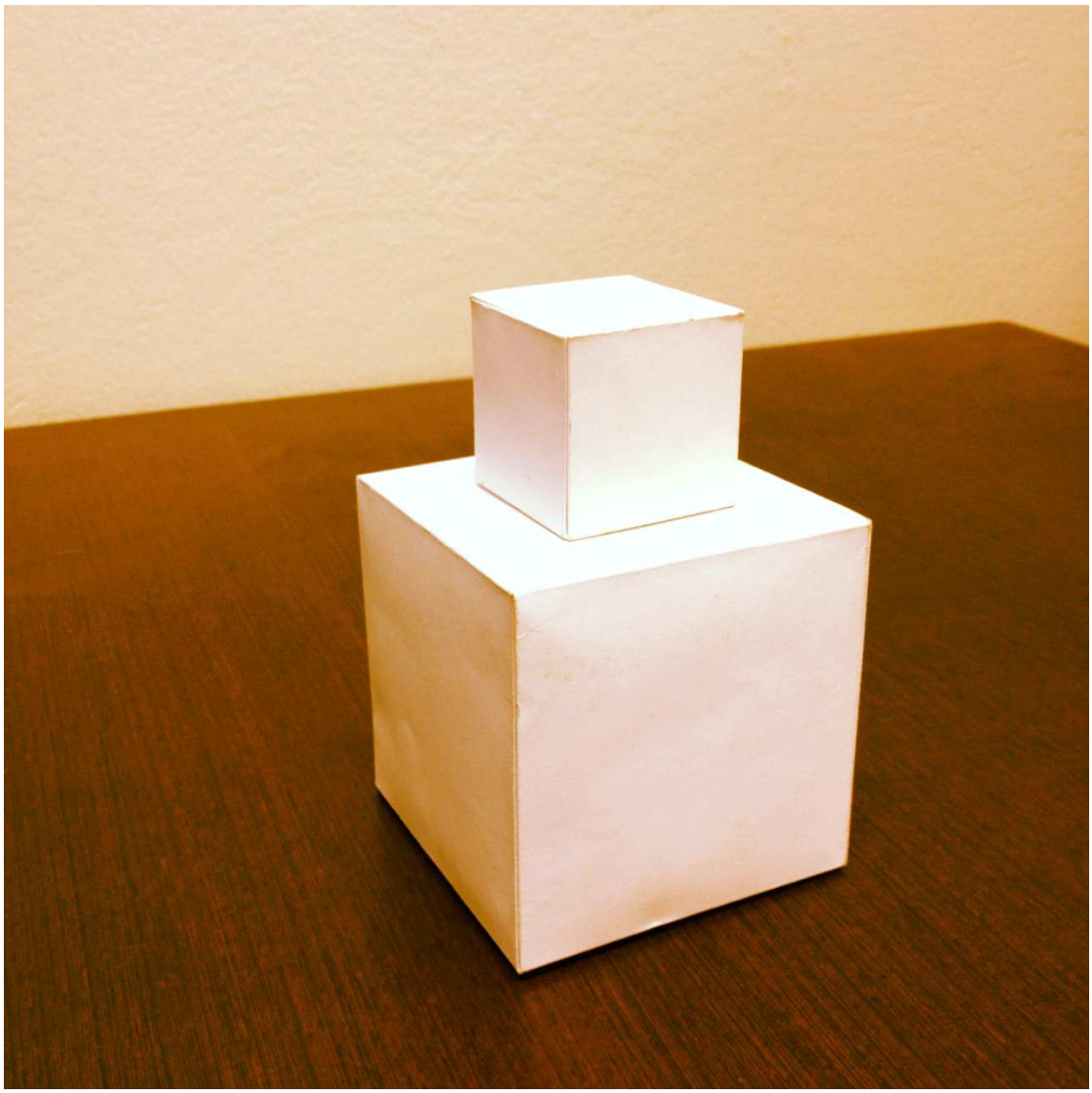
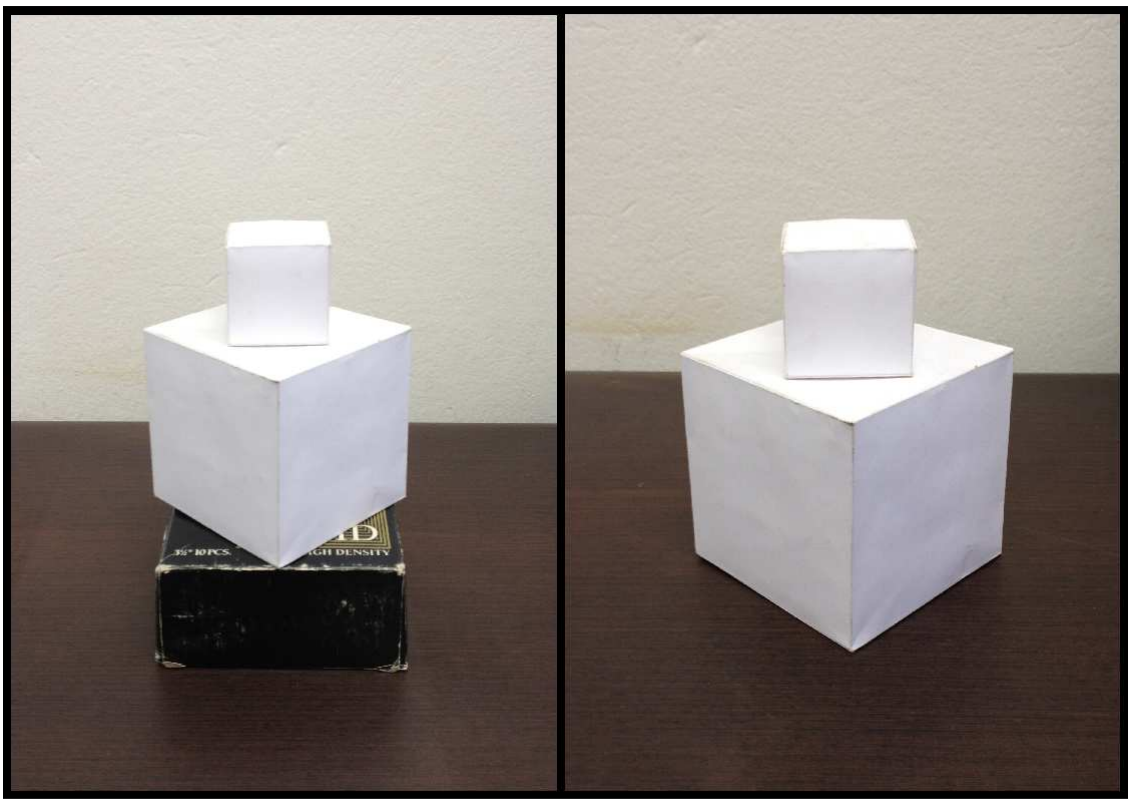


Figura 4.2 Caixas de papel como objetos para gerar as nuvens de pontos.

Duas imagens foram utilizadas para fazer o registro, com a ajuda das caixas da Figura 4.2. Na primeira imagem as caixas foram empilhadas sobre outro objeto, como mostra a Figura 4.3 (a). Neste trabalho esta imagem será chamada de “objeto1”. Na segunda, as caixas foram empilhadas sobre uma mesa como mostra a Figura 4.3 (b). Neste trabalho esta imagem será chamada de “objeto2”.



(a) (b)

Figura 4.3 Par de imagens utilizadas para o registro. (a) Objeto1. (b) Objeto2.

O processamento das imagens foi feito utilizando o *Kinetic*® (ver Figura 4.4) dispositivo utilizado para jogos da console chamada *X-BOX 360*®. Esse é um novo sistema de controle da *Microsoft*® que faz parte do projeto *Natal*[1], projeto que utiliza um digitalizador de luz estruturada para reconhecer os movimentos dos jogadores no sistema, servindo como sensor de movimento.

[1] Projeto realizado pela Microsoft® para o desenvolvimento de um sensor de movimento de última geração para o console de videogame X-Box.

Figura 4.4 Imagem do *Kinectic*.

Esse dispositivo foi utilizado para capturar imagens em um formato chamado "*Polygon File Format*" (*.ply*). Esse formato foi desenvolvido principalmente para armazenar dados tridimensionais de scanners *3-D*. Esse formato permite o armazenamento de diferentes propriedades do polígono tanto na vista frontal quanto traseira, além de armazenar cor, transparência, normais à superfície, coordenadas de textura e valores de confiança. Neste trabalho só se utilizaram coordenadas de posição, utilizando o programa "*kinect-3dscanner.exe*" (ver Figura 4.5).

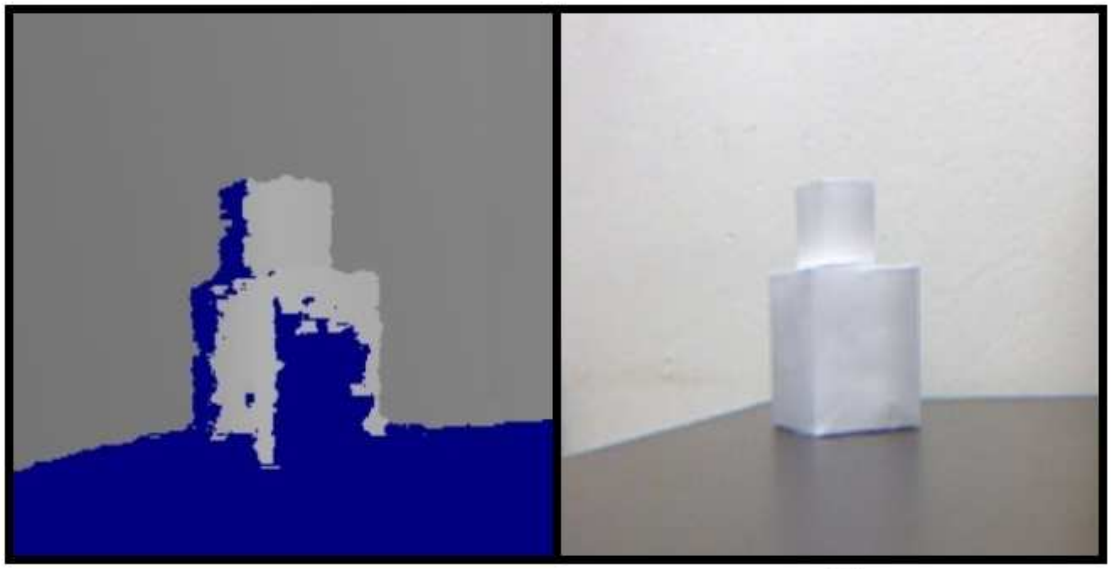


Figura 4.5 (a) Interface gráfica do programa "*kinect-3dscanner.exe*". (b) Vista da câmera.

Depois de obter as imagens neste formato (*.ply*) foi preciso transformar as imagens para outro formato "*textfile*" (*.txt*), o qual é um formato capaz de guardar a informação das coordenadas de todos os pontos que compõem as nuvens de pontos, já que permite armazenar o conjunto de pontos tridimensionais no do sistema, definindo as coordenadas como "X", "Y" e "Z" com a função de representar as superfícies externas do objeto. Para transformar as imagens do formato "*.ply*" ao formato "*.txt*" foi necessário utilizar o programa "*3D Object Converter*" (ver Figura 4.6), *software* livre, muito útil para conseguir observar as imagens "*.ply*" e posteriormente transformá-las ao formato "*.txt*" como amostra a Figura 4.7.

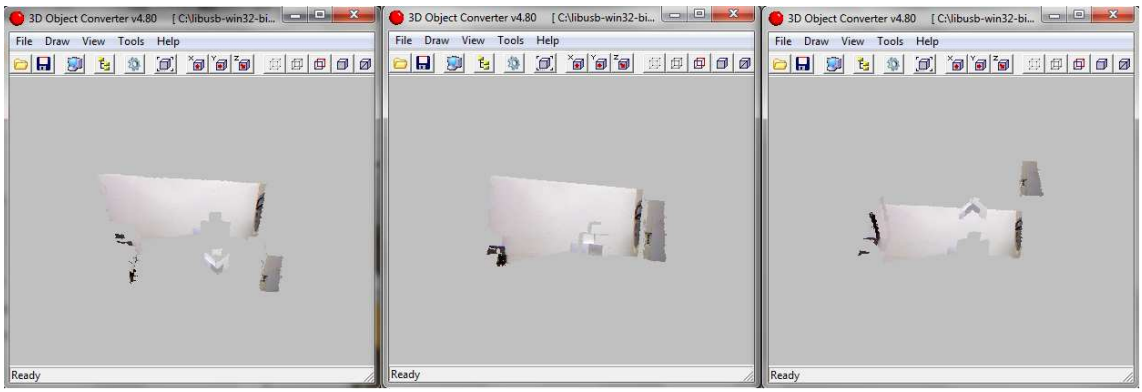


Figura 4.6 Interface gráfica do programa "*3D Object Converter*".

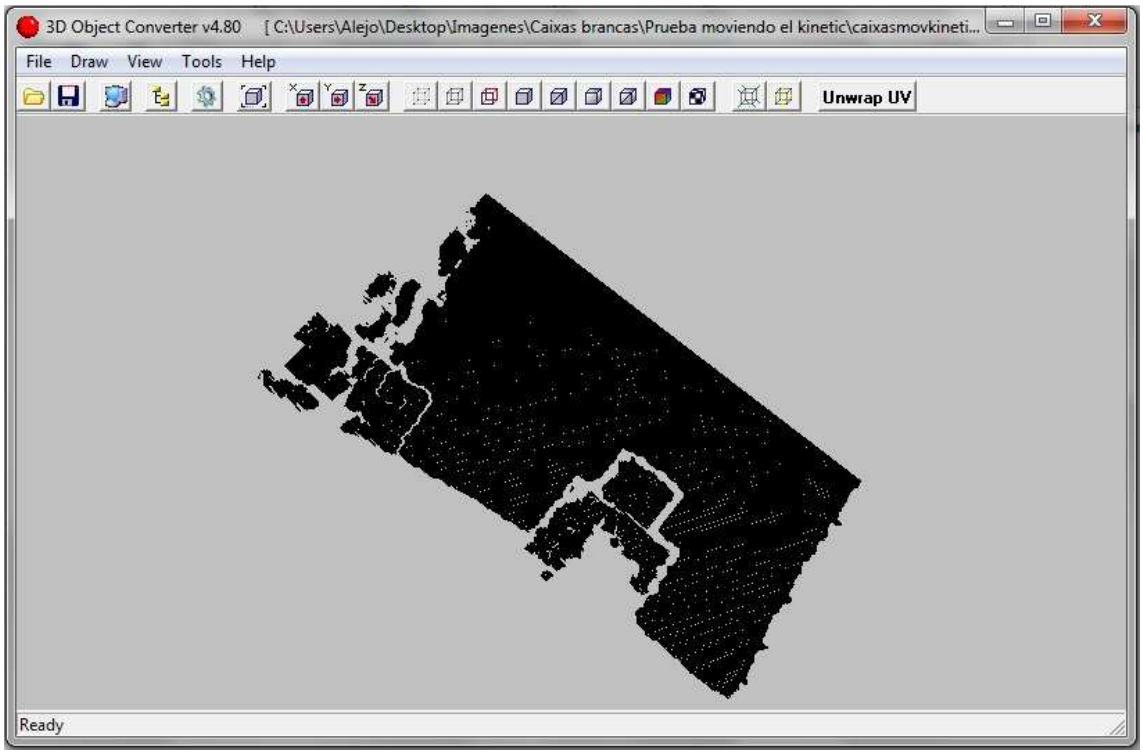


Figura 4.7 Nuvem de pontos obtida da transformação do formato "*.ply*" no formato "*.txt*".

A nuvem de pontos no formato "*.txt*" foi "polida" tirando dados que não fizeram parte do objetivo do estudo. Para isso, utilizou-se o programa "*MeshLab*" (*software* livre) obtendo a nuvem de pontos específico como pode-se ver na Figura 4.8.

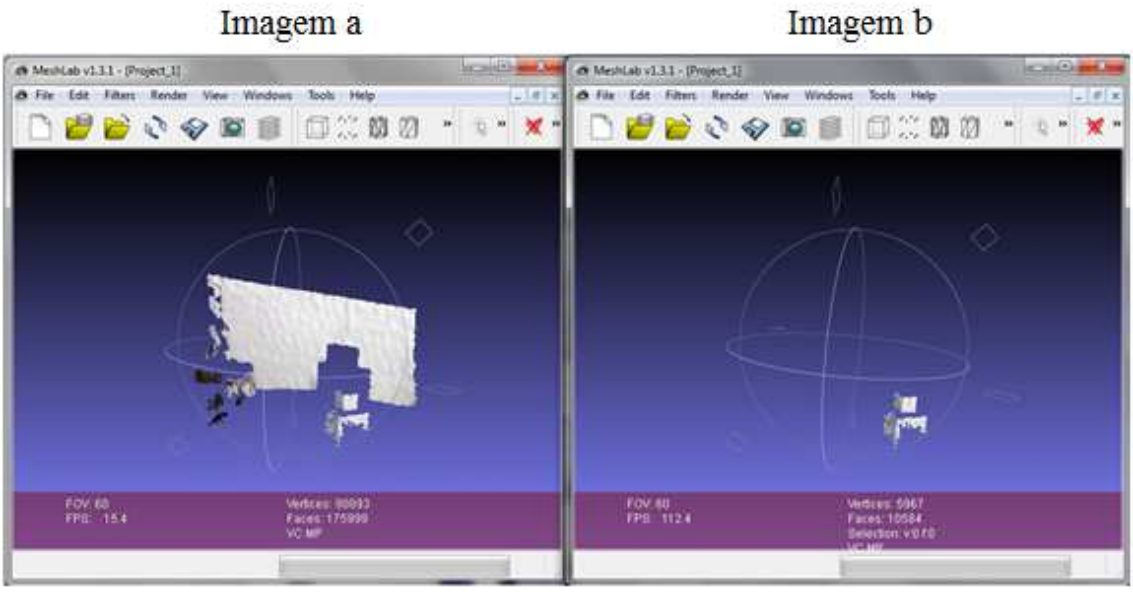


Figura 4.8 Imagem a-).*ply* sem polir. Imagem b-).*ply* polida.

Como resultado, foram obtidas as nuvens de pontos utilizadas neste projeto (ver Figura 4.9).

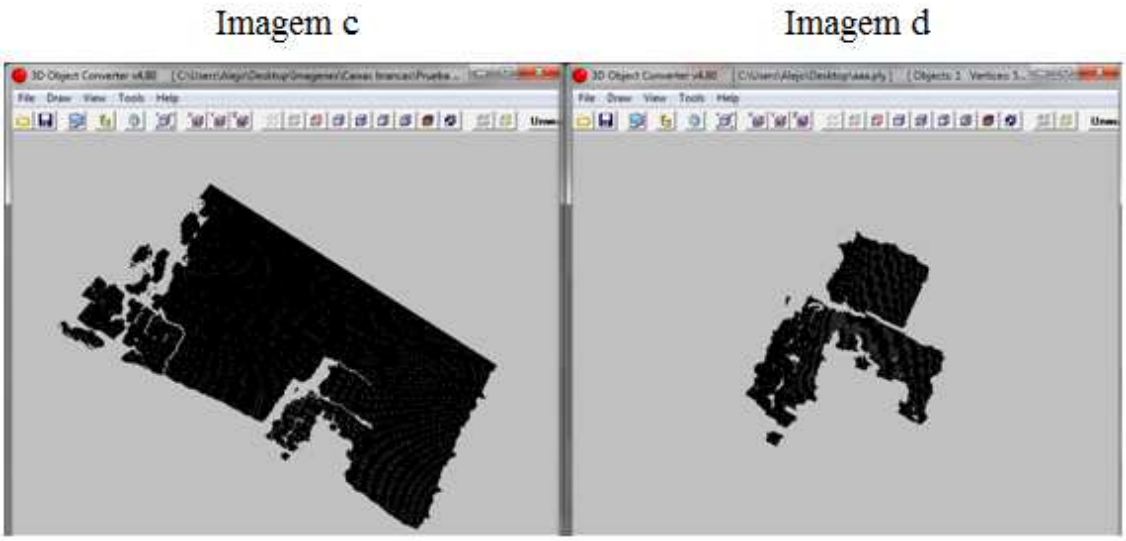


Figura 4.9 Imagem c-) Nuvem de pontos completa. Imagem d-) Nuvem de pontos do estudo.

Assim, foram obtidas as imagens para realizar os testes utilizando os algoritmos implementados.

4.2 EQUIPAMENTO

As especificações técnicas do digitalizador de luz estruturada utilizado (*Kinectic*), com base em cada uma das características são:

- Sensor

    - Lentes com detecção de cores e profundidade;

    - Microfone de voz;

    - Motor de inclinação para ajuste do sensor;

- Campo de visão;

    - Campo de visão horizontal: 57 graus;

    - Campo de visão vertical: 43 graus;

    - Alcance físico da inclinação: ± 27 graus;

    - Alcance do sensor de profundidade: 1.2m - 3.5m (metros);

- Fluxo de dados;

    - 320x240 16-*bit depth* a 30*FPS* (*Frames Per Second*);

    - 640x480 32-*bit colour* a 30*FPS* (*Frames Per Second*);

    - 16-*bit audio @* 16 *kHz* (Kilo*Hertz*).

Muitas das características dos algoritmos também dependem da linguagem de programação escolhida para desenvolvê-los. Como neste trabalho foram aplicadas múltiplas operações, além de cômputos matriciais para o armazenamento e manipulação de dados *3-D* através de matrizes multidimensionais, foi escolhido o ambiente de programação *Matlab* ® para solucionar de maneira ágil os problemas na programação de algoritmos robustos além de oferecer uma interface gráfica ótima para visualizar os resultados. Todos os algoritmos neste trabalho foram desenvolvidos utilizando *Matlab* ®.

4.3 SIMULAÇÃO E IMAGENS OBTIDAS

O *ICP* é um algoritmo que depende da crítica visual para provar sua convergência, ou seja, pode-se verificar a convergência do algoritmo vendo o resultado entre as duas imagens utilizadas, mas não existe uma garantia teórica da convergência do algoritmo para alguns valores exatos. Todos os dados obtidos são renderizados (visualização *2-D* de dados em *3-D*).

Os testes com os dados das nuvens de pontos foram feitos da seguinte maneira:

I. Teste 1: Renderização obtida com o algoritmo *ICP* “forca_bruta”.

II. Teste 2: Renderização obtida com o algoritmo *ICP* “kd_tree”.

III. Teste 3: Alteração do número de iterações.

IV. Teste 4: Velocidade de convergência das imagens implementando ruído gaussiano sobre uma das imagens.

V. Teste 5: Gráficos do erro (RMS) *versus* o tempo.

Todos os testes foram realizados com 10 iterações do *ICP*, menos o Teste 3 no qual se fizeram provas para 1, 50 e 100 iterações.

4.3.1 Teste 1: Renderização obtida com o algoritmo *ICP* “força bruta”

Inicialmente foram feitas simulações utilizando o *ICP*, construído utilizando o algoritmo de busca “força bruta”, no qual, considerando duas imagens (P e Q) cada uma com um conjunto de pontos (p_i e q_i), são procurados pontos correspondentes calculando a distância entre um ponto da imagem “P” com outro ponto na imagem “Q”. A renderização da nuvem de pontos obtida pelas nuvens de pontos iniciais também foi registrada.

A vista frontal é a vista plana dos eixos x e y. No Gráfico 1 da Figura 4.10 observa-se a imagem frontal da nuvem de pontos obtida do objeto 1. No Gráfico 2 da mesma figura observa-se a imagem frontal da segunda nuvem de pontos (objeto 2). No Gráfico 3 observa-se a imagem frontal do registro obtido com o algoritmo *ICP* “força bruta”. E no Gráfico 4 uma imagem *3-D* do alinhamento. Nos Gráficos o objeto 1 é representado com a cor amarela e o objeto 2 com a cor azul.

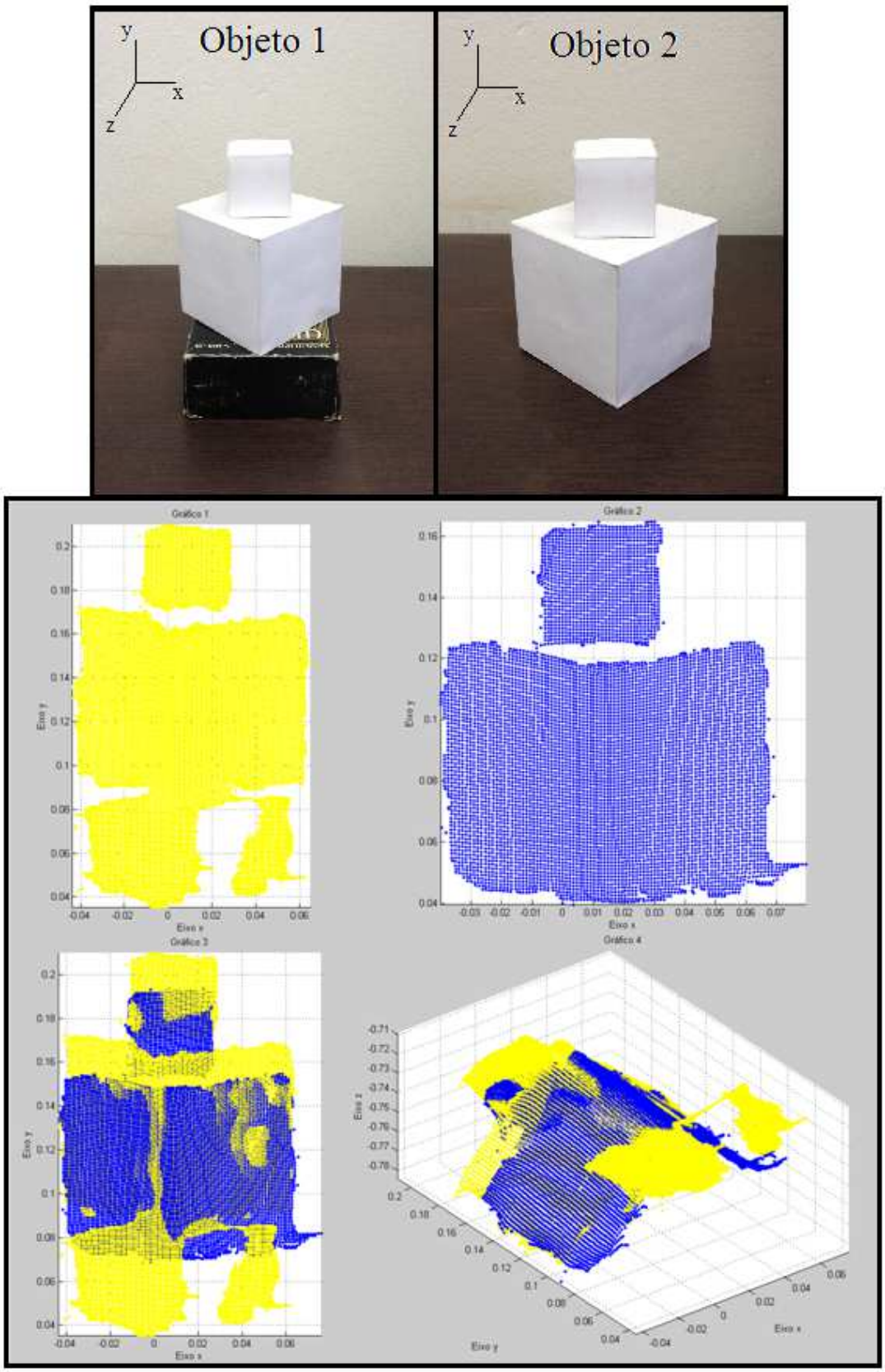


Figura 4.10 Vista frontal das imagens e do registro obtido pelo algoritmo *ICP* "força bruta" com 10 iterações. Gráfico 1, vista frontal do objeto 1. Gráfico 2, vista frontal do objeto 2. Gráfico 3, vista frontal do registro entre os dois objetos. Gráfico 4, imagem perspectiva do registro.

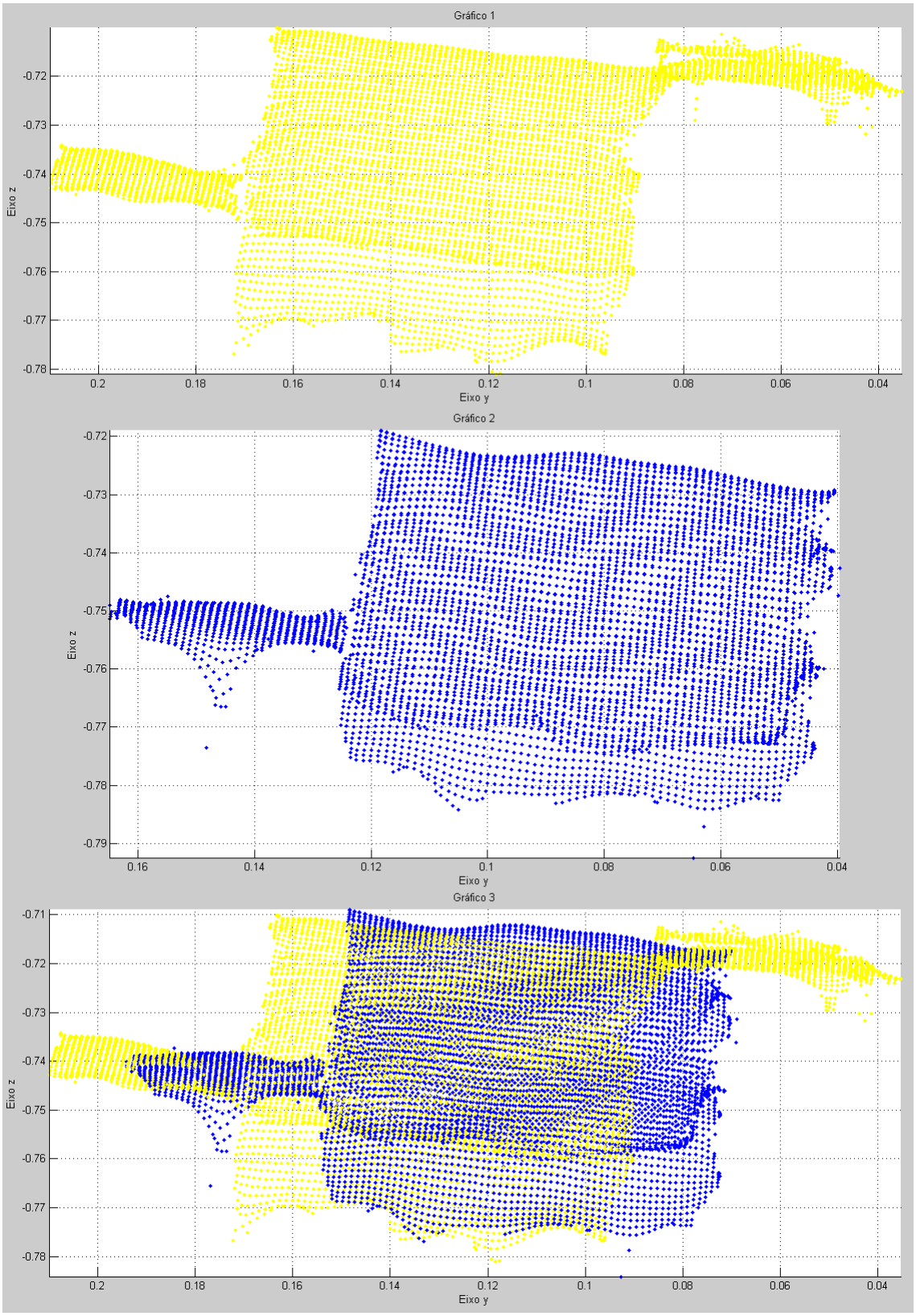


Figura 4.11 Vista lateral das imagens e do registro obtido pelo algoritmo *ICP* “força bruta” com 10 iterações. Gráfico 1, vista lateral do objeto 1. Gráfico 2, vista lateral do objeto 2. Gráfico 3, vista lateral do registro entre os dois objetos.

A vista lateral é a vista plana dos eixos z e y. No Gráfico 1 da Figura 4.11 observa-se a imagem lateral da nuvem de pontos obtida do primeiro objeto. No Gráfico 2 da mesma figura observa-se a imagem lateral da segunda nuvem de pontos (segundo objeto). No Gráfico 3 observa-se a imagem lateral do registro obtido com o algoritmo *ICP* "força bruta".

4.3.2 Teste 2: Renderização obtida com o algoritmo *ICP "kd tree"*

Este teste foi realizado utilizando o algoritmo *ICP* com a adaptação do algoritmo de busca "*kd tree*". Esse tipo de algoritmo foi implementado devido à diminuição do tempo de busca dos pontos correspondentes entre as duas imagens, gerando um custo computacional mais baixo e sendo mais rápido que o *ICP* "força bruta".

A vista frontal é a vista plana dos eixos x e y. No Gráfico 1 da Figura 4.12 observa-se a imagem frontal da nuvem de pontos obtida do objeto 1. No Gráfico 2 da mesma figura observa-se a imagem frontal da segunda nuvem de pontos (objeto 2). No Gráfico 3 observa-se a imagem frontal do registro obtido com o algoritmo *ICP* "*kd tree*". E no Gráfico 4 uma imagem *3-D* do alinhamento.Nos Gráficos o objeto 1 é representado com a cor amarela e o objeto 2 com a cor azul.

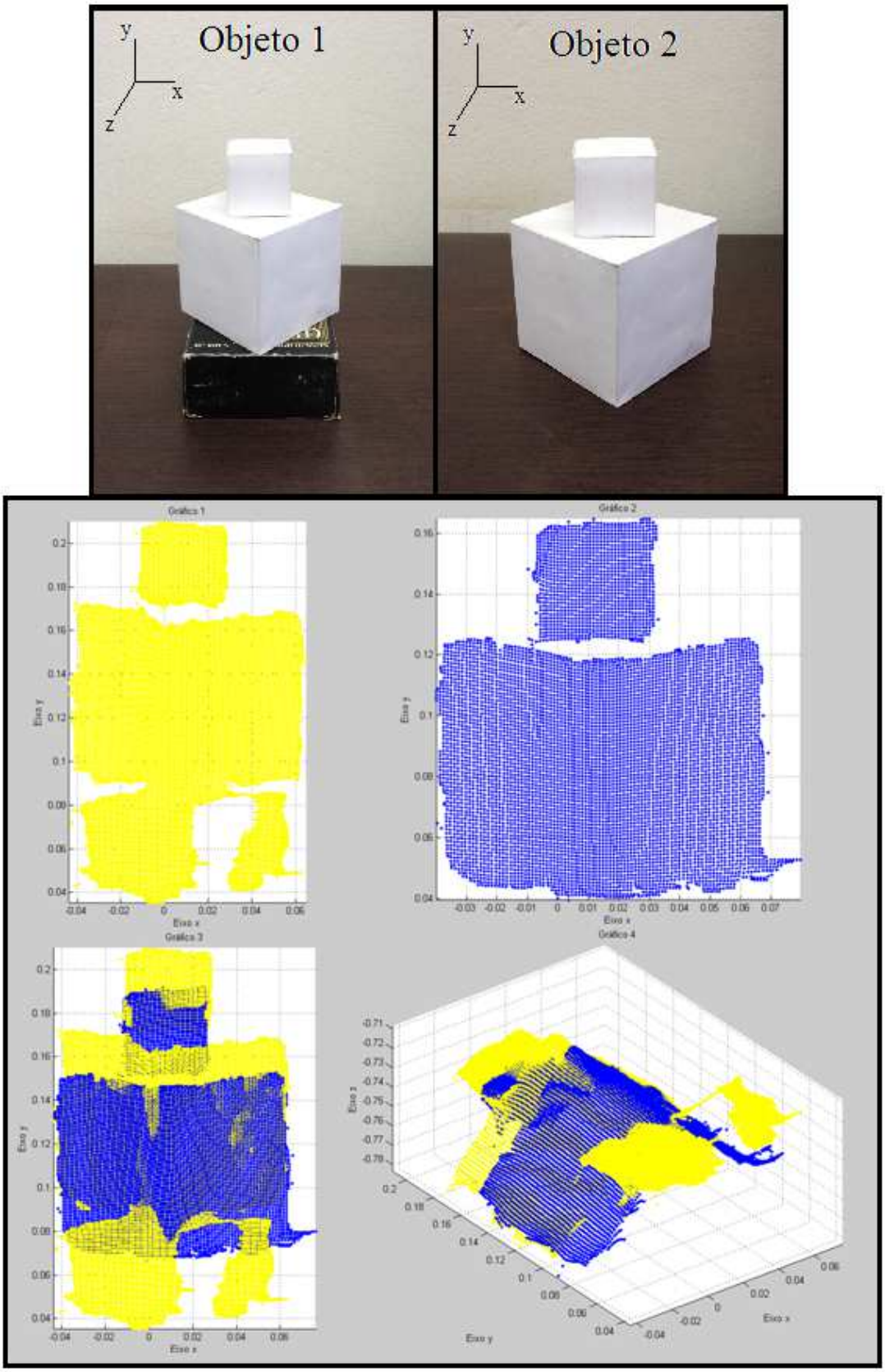


Figura 4.12 Vista frontal das imagens e do registro obtido pelo algoritmo *ICP* "*kd tree*" com 10 iterações. Gráfico 1, vista frontal do objeto 1. Gráfico 2, vista frontal do objeto 2. Gráfico 3, vista frontal do registro entre os dois objetos. Gráfico 4, imagem perspectiva do registro.

A vista lateral é a vista plana dos eixos z e y. No Gráfico 1 da Figura 4.13 observa-se a imagem lateral da nuvem de pontos obtida do primeiro objeto. No Gráfico 2 da mesma figura observa-se a imagem lateral da segunda nuvem de pontos (segundo objeto). No Gráfico 3 observa-se a imagem lateral do registro obtido com o algoritmo *ICP* "*kd tree*".

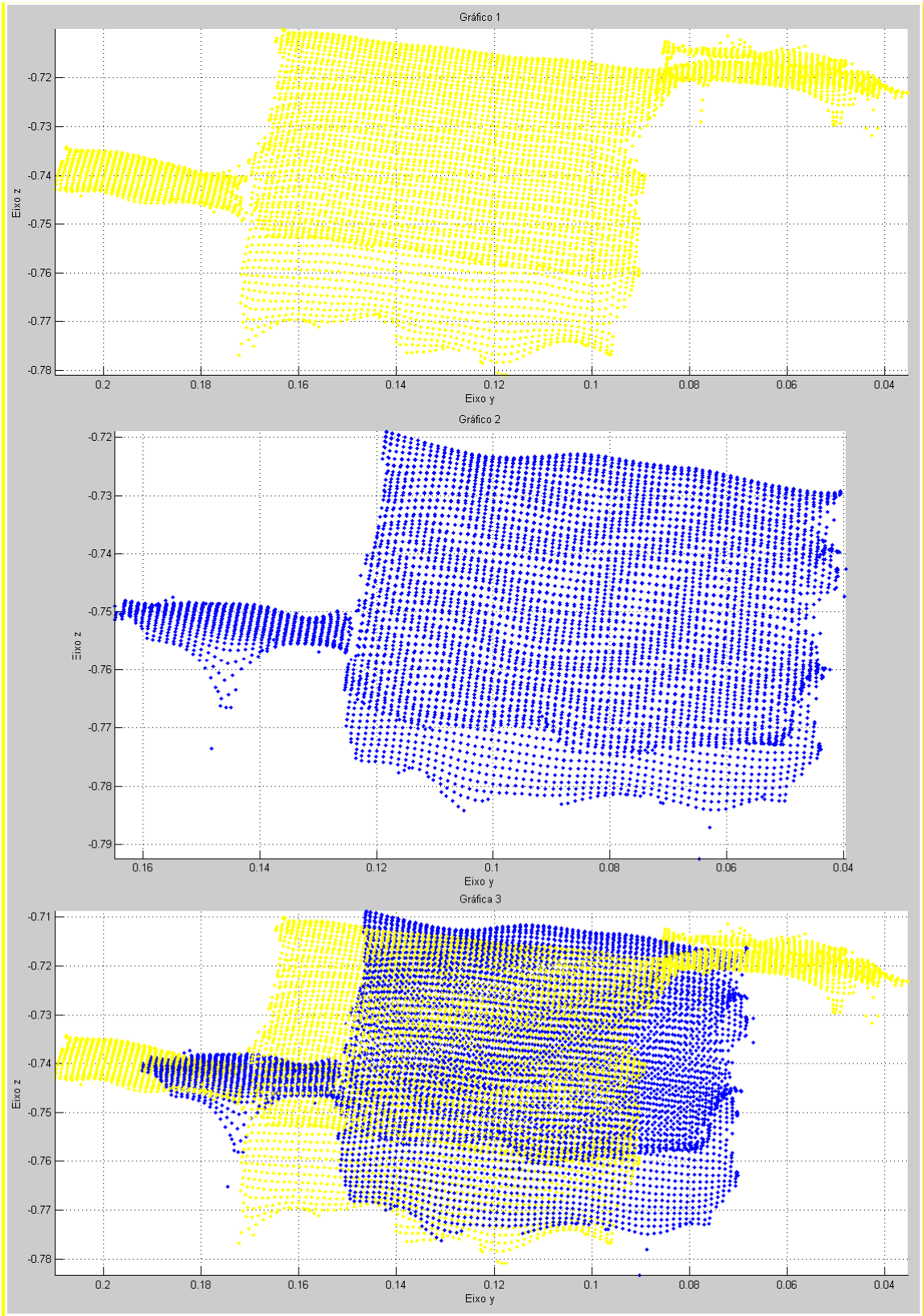


Figura 4.13 Vista lateral das imagens e do registro obtido pelo algoritmo *ICP* "*kd tree*"com 10 iterações. Gráfico 1, vista lateral do objeto 1. Gráfico 2, vista lateral do objeto 2. Gráfico 3, vista lateral do registro entre os dois objetos.

4.3.3 Teste 3: Variação do número de iterações

Foram feitos testes com os dois tipos de algoritmos “força bruta” e “*kd tree*”, cada um com 1, 50 e 100 iterações, nas Figuras 4.14, 4.15, 4.16, 4.17, 4.18 e 4.19. Os gráficos 1, 2 e 3 correspondem respectivamente às vistas frontais, à vista perspectiva, e às vistas laterais do registro entre o primeiro objeto e o segundo objeto para cada uma das iterações.

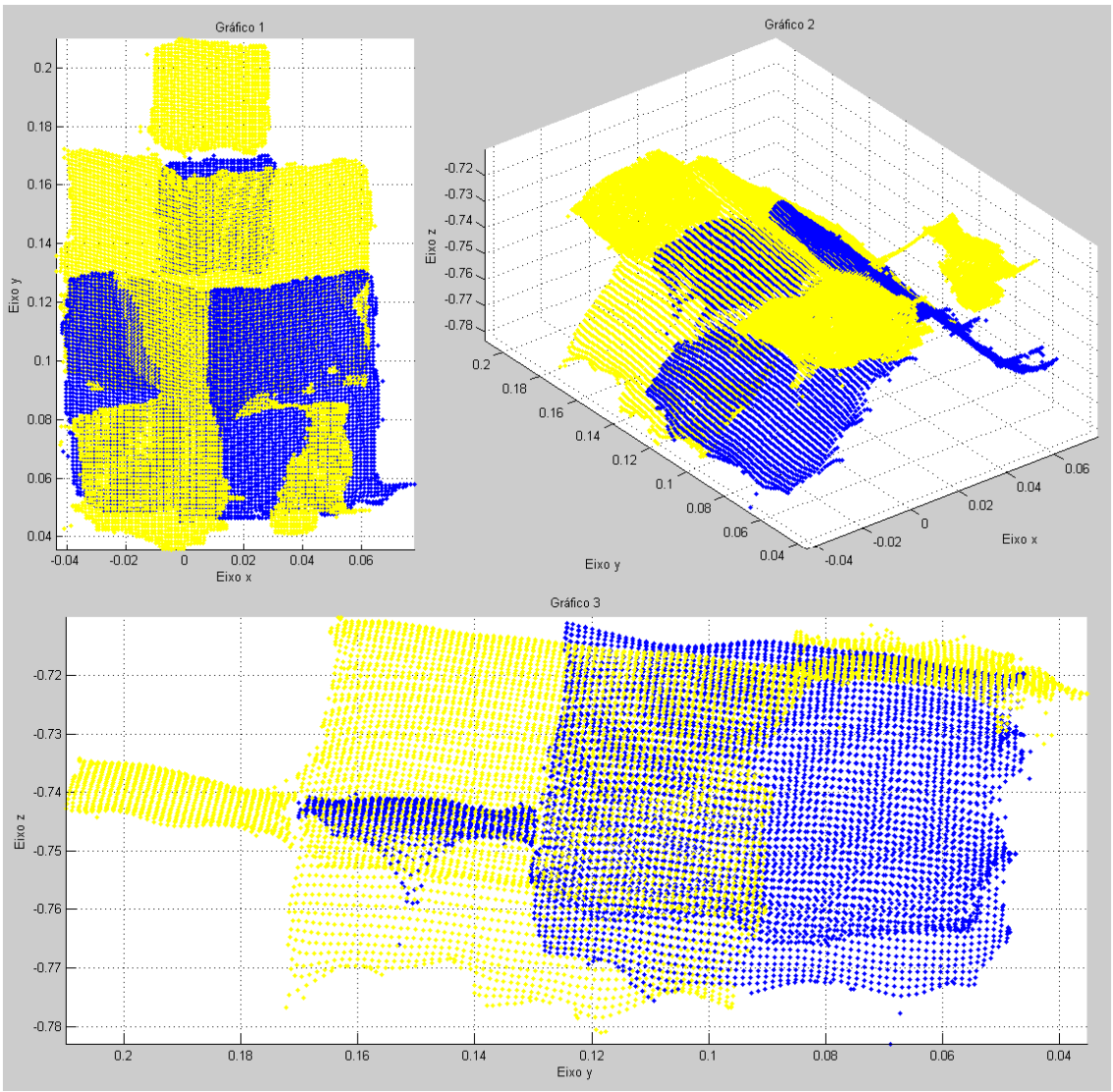


Figura 4.14 *ICP* “força bruta” com 1 iteração.

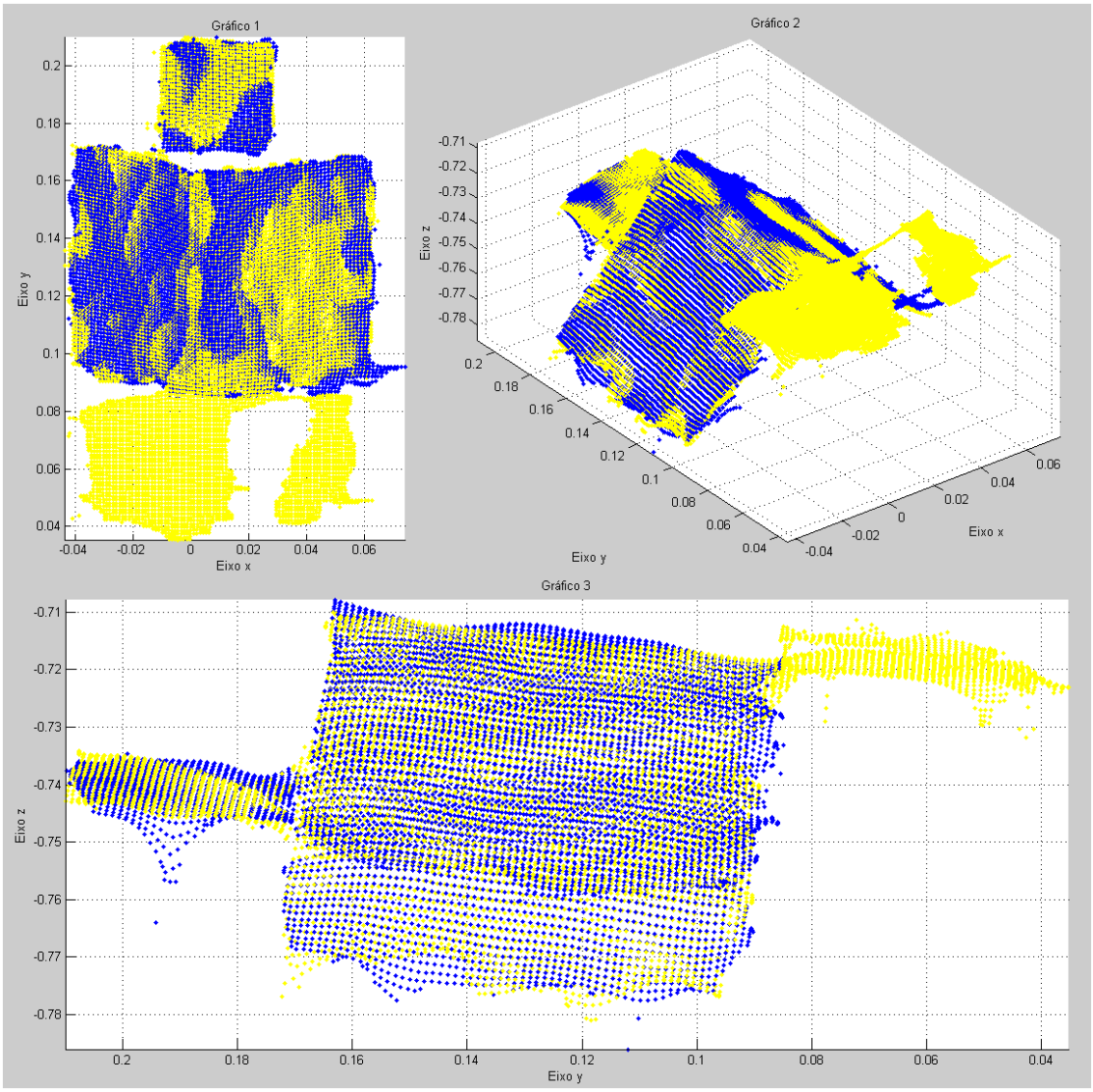


Figura 4.15 *ICP* “força bruta” com 50 iterações.

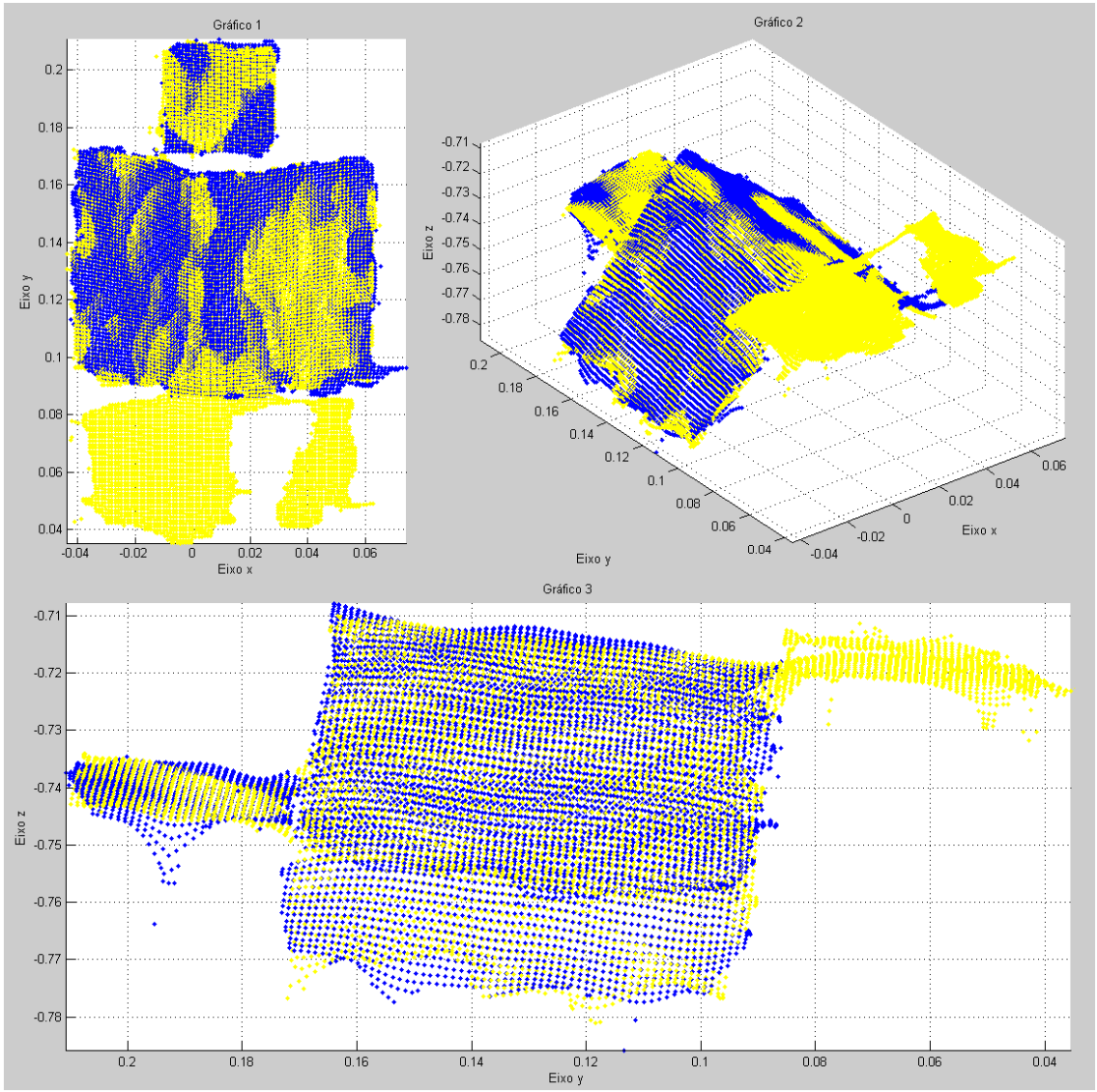


Figura 4.16 *ICP* “força bruta” com 100 iterações.

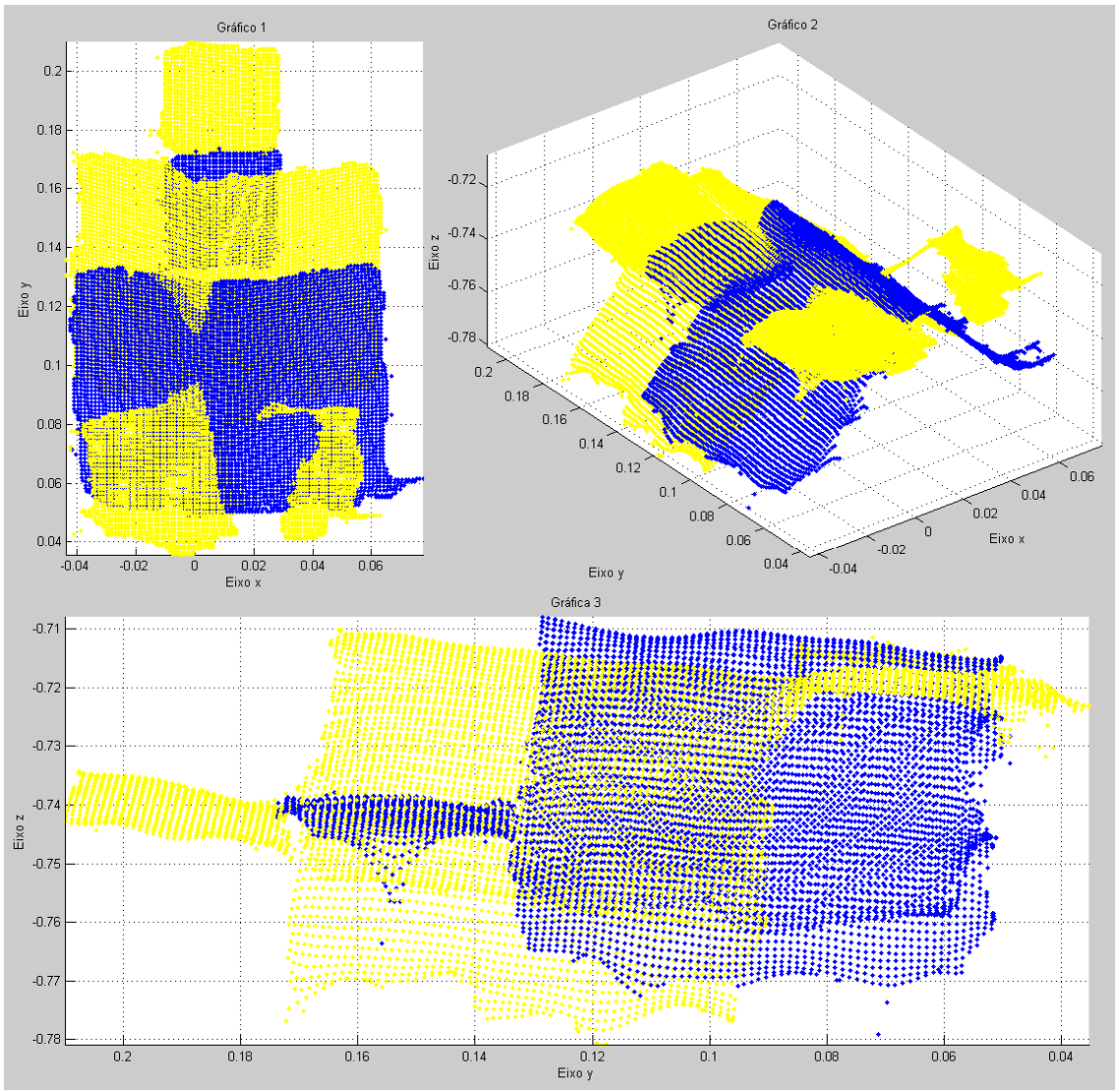


Figura 4.17 *ICP* "*kd tree*" com 1 iteração.

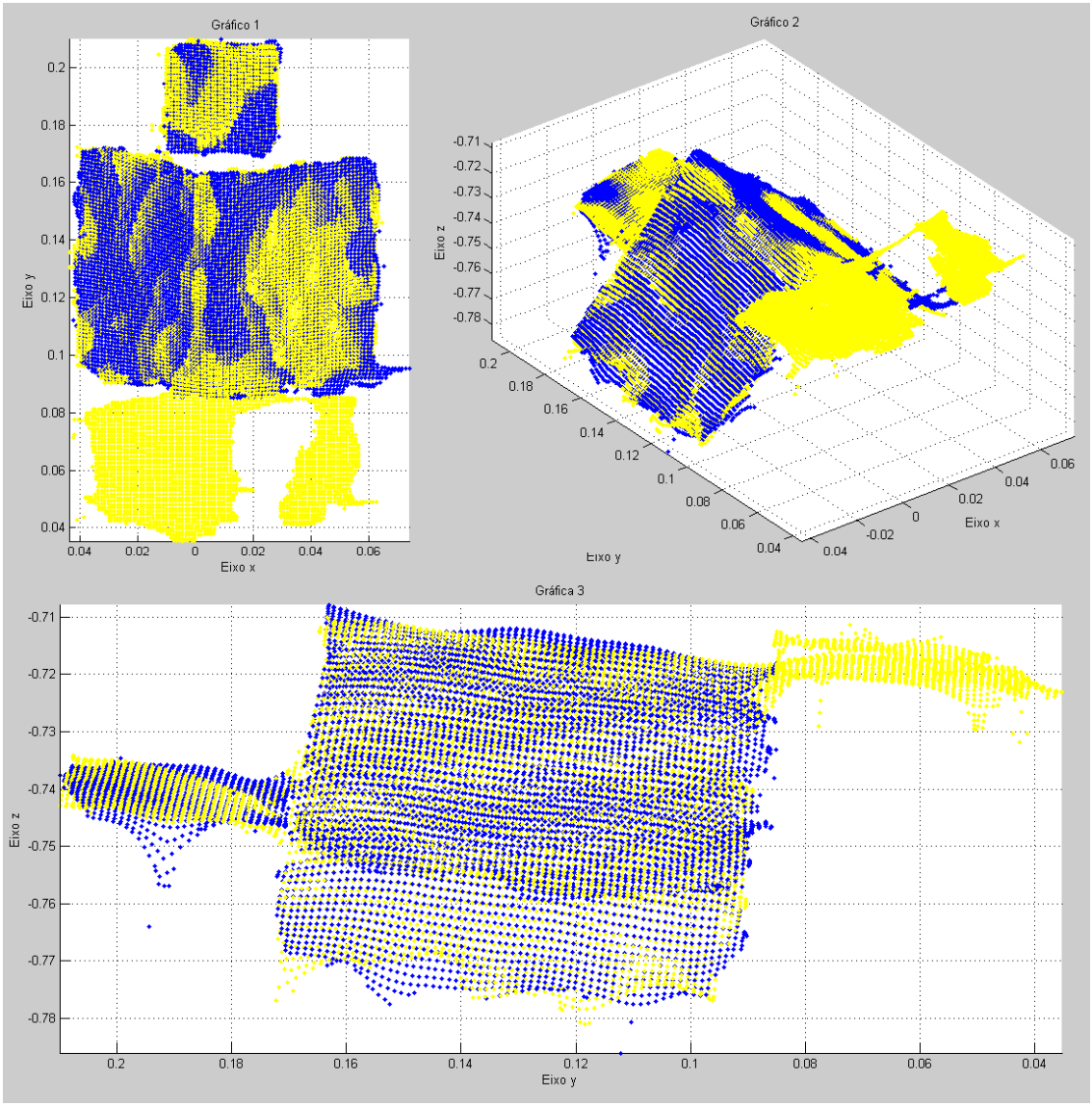


Figura 4.18 *ICP* "*kd tree*" com 50 iterações.

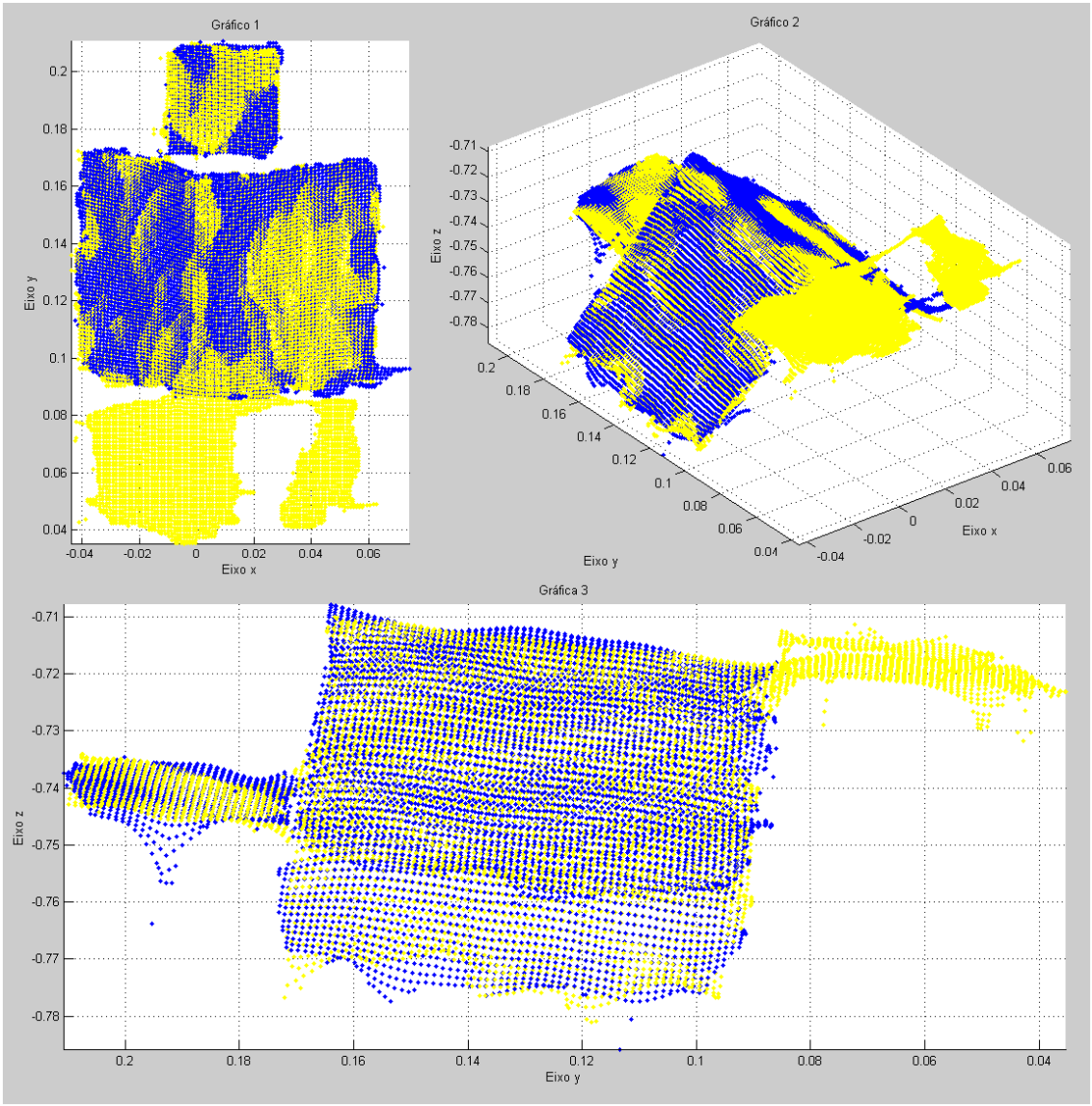


Figura 4.19 *ICP* "*kd tree*" com 100 iterações.

4.3.4 Teste 4: Velocidade de convergência das imagens implementando ruído gaussiano sobre uma das imagens

Foi adicionado ruído gaussiano sobre o objeto 2 nos algoritmos para uma das duas imagens e foram obtidas as imagens do registro e do valor quadrático médio para os dois tipos de algoritmo *ICP* (ver Figuras 4.20, 4.21, 4.22, 4.23, 4.24, e 4.25). O desvio padrão implementado foi 10^{-2} para 1, 10 e 100 iterações. O objetivo de adicionar ruído é encontrar se as variações feitas com o ruído dentro do objeto atrapalham o processo de registro de

nuvens de pontos, para demonstrar a robustez do algoritmo.

Os gráficos 1, 2, 3 e 4 correspondem respectivamente à vista lateral do objeto 1, à vista lateral do objeto 2, à vista lateral do registro entre os dois objetos e ao calculo do *RMS* para cada um dos níveis de ruído.

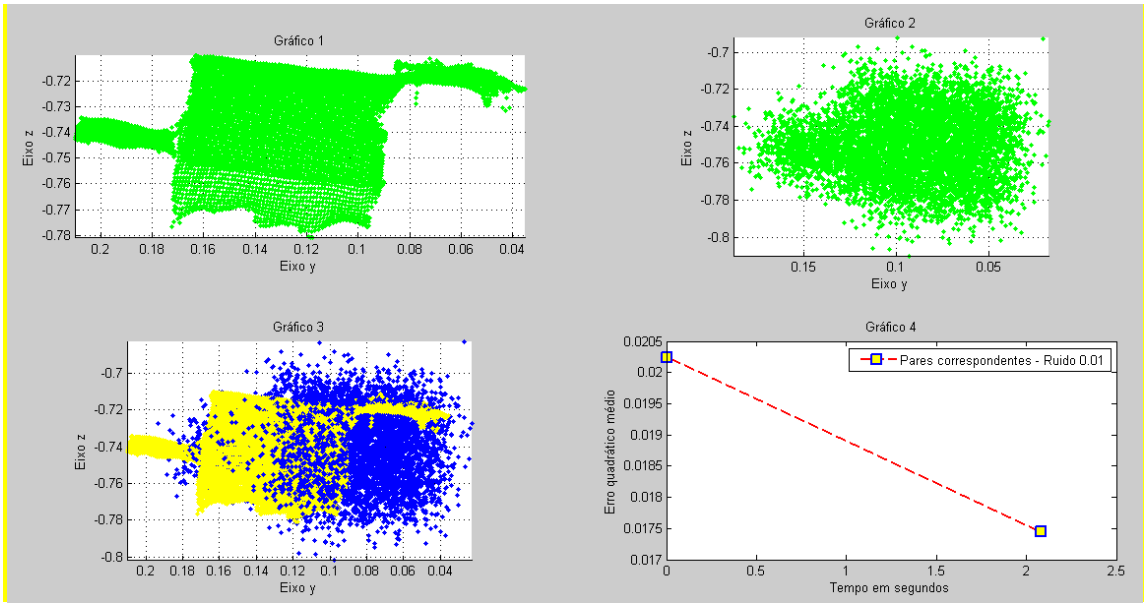


Figura 4.20 Imagens obtidas aplicando um ruído Gaussiano sobre o objeto 2 no algoritmo *ICP* "força bruta" com um desvio padrão de 10^{-2} com 1 iteração.

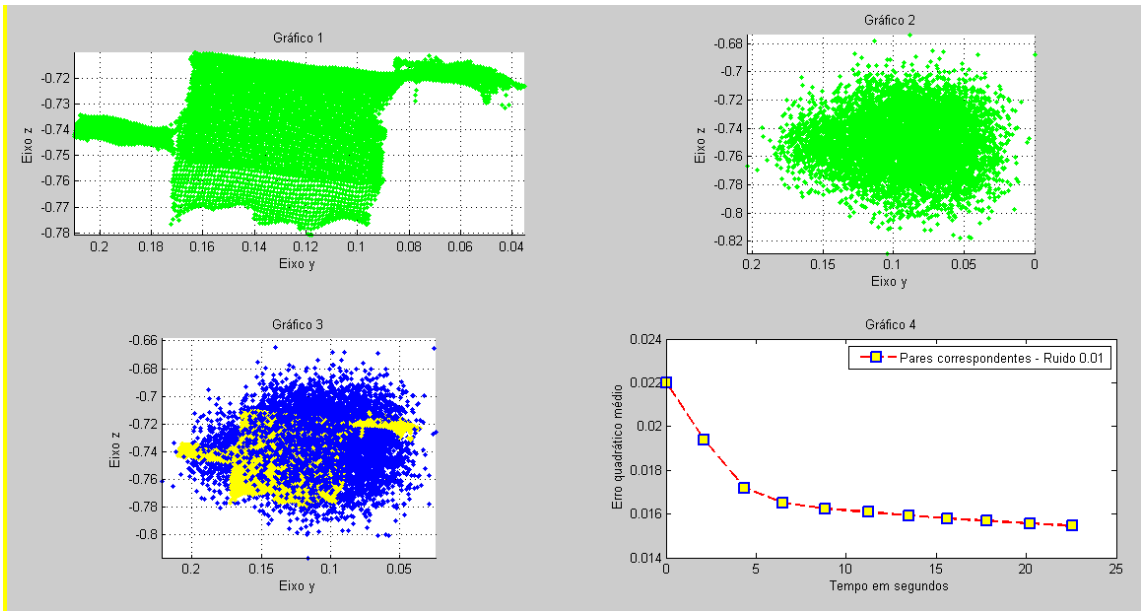


Figura 4.21 Imagens obtidas aplicando um ruído Gaussiano sobre o objeto 2 no algoritmo *ICP* "força bruta" com um desvio padrão de 10^{-2} com 10 iterações.

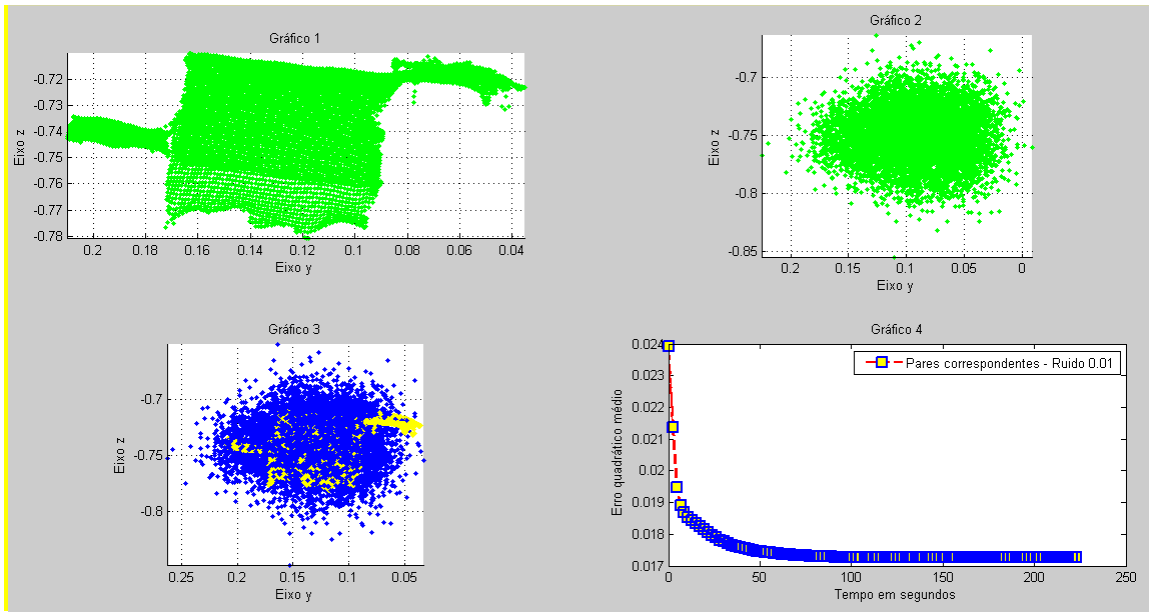


Figura 4.22 Imagens obtidas aplicando um ruído Gaussiano sobre o objeto 2 no algoritmo *ICP* "força bruta" com um desvio padrão de 10^{-2} com 100 iterações.

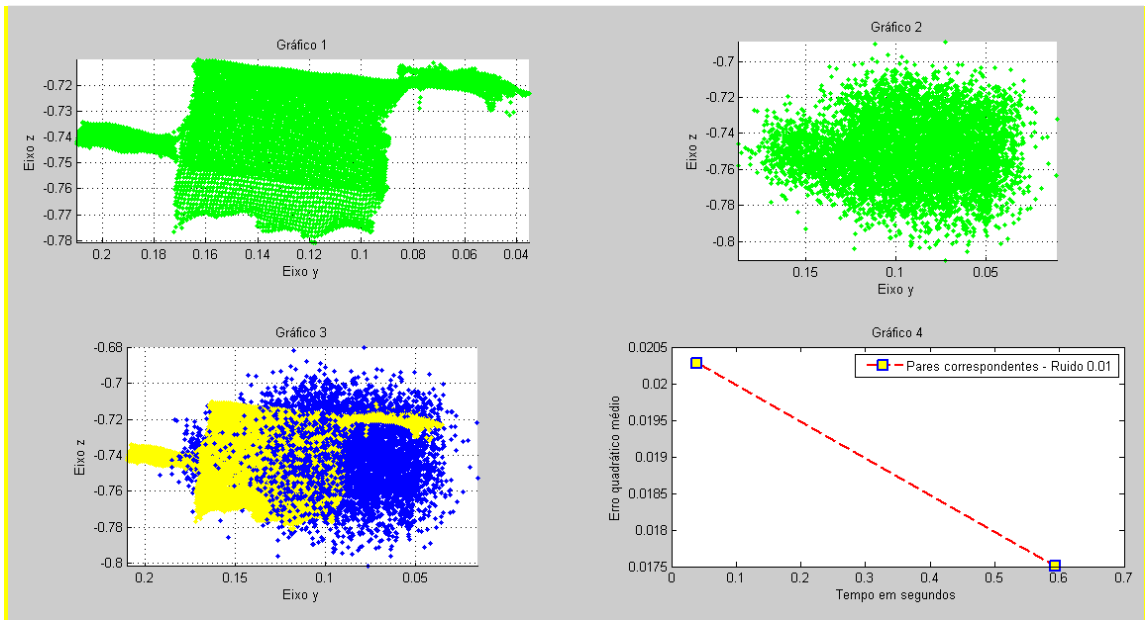


Figura 4.23 Imagens obtidas aplicando um ruído Gaussiano sobre o objeto 2 no algoritmo *ICP* "*kd tree*" com um desvio padrão de 10^{-2} com 1 iteração.

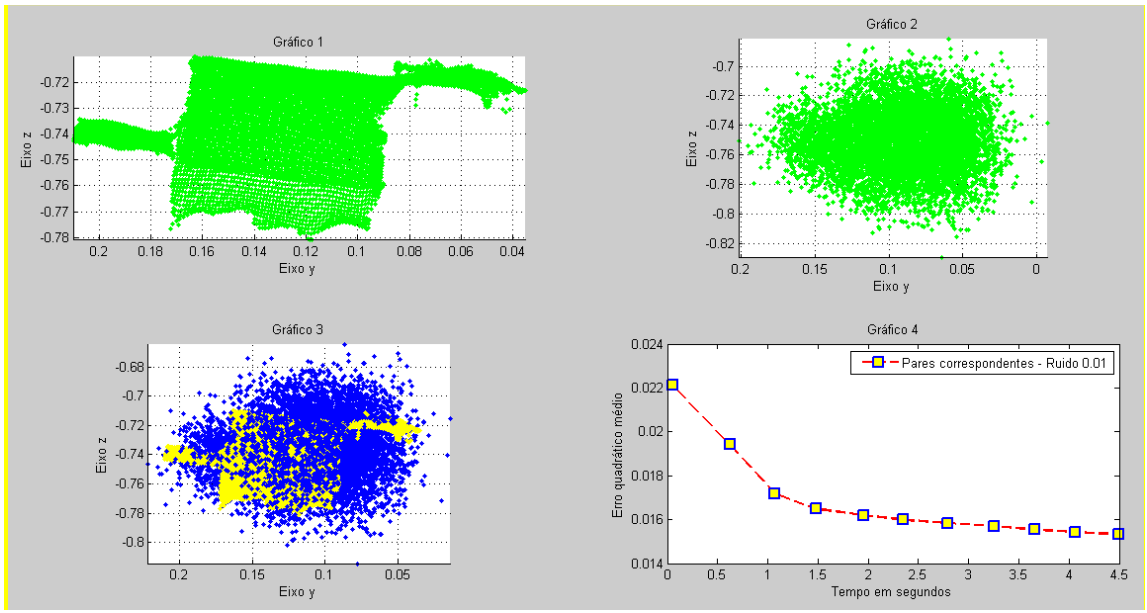


Figura 4.24 Imagens obtidas aplicando um ruído Gaussiano sobre o objeto 2 no algoritmo *ICP* "*kd tree*" com um desvio padrão de 10^{-2} com 10 iterações.

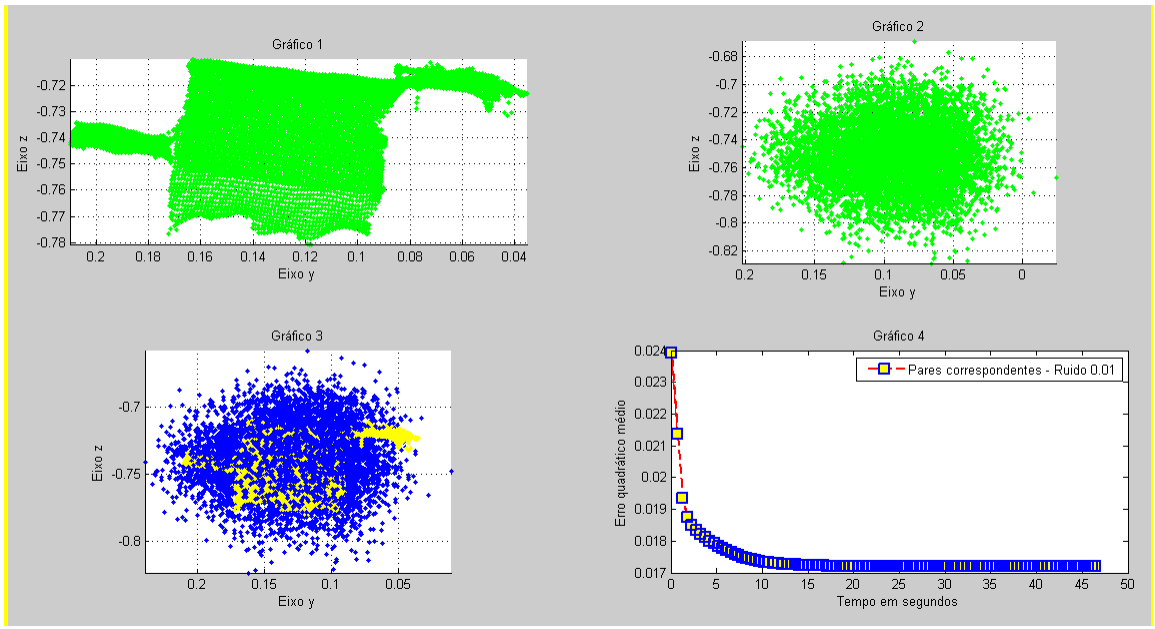


Figura 4.25 Imagens obtidas aplicando um ruído Gaussiano sobre o objeto 2 no algoritmo *ICP* "*kd tree*" com um desvio padrão de 10^{-2} com 100 iterações.

4.3.5 Teste 5: Gráficos do erro (RMS) versus o tempo

O erro quadrático médio foi calculado para os algoritmos do *ICP* "força bruta" e "*kd tree*"

com diferentes números de iterações sem ruído, como mostram as Figuras 4.26, 4.27, 4.28, 4.29, 4.30, 4.31, 4.32, e 4.33.

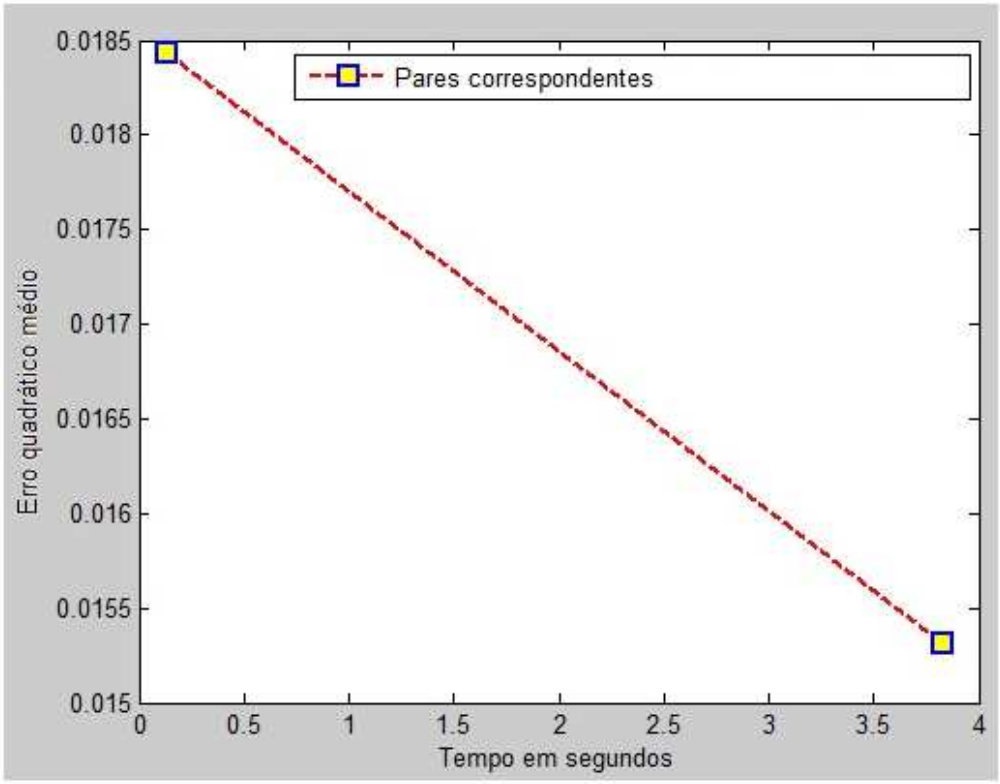


Figura 4.26 Erro quadrático médio obtido utilizando o algoritmo *ICP* “força bruta” com 1 iteração.

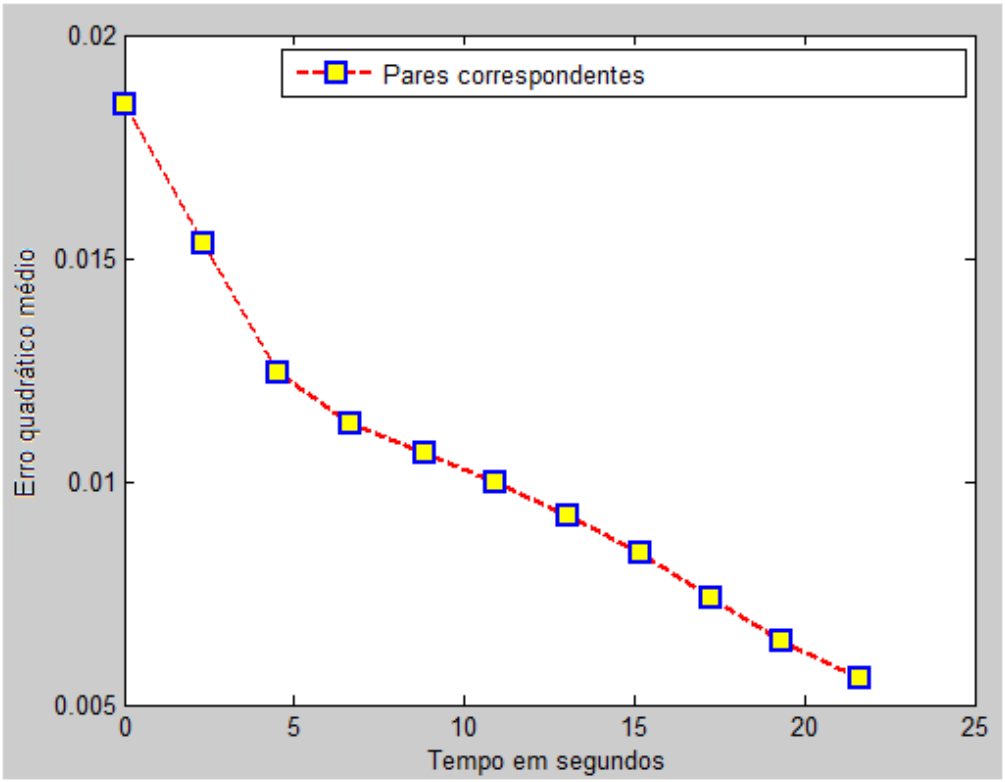


Figura 4.27 Erro quadrático médio obtido utilizando o algoritmo *ICP* "força bruta" com 10 iterações.

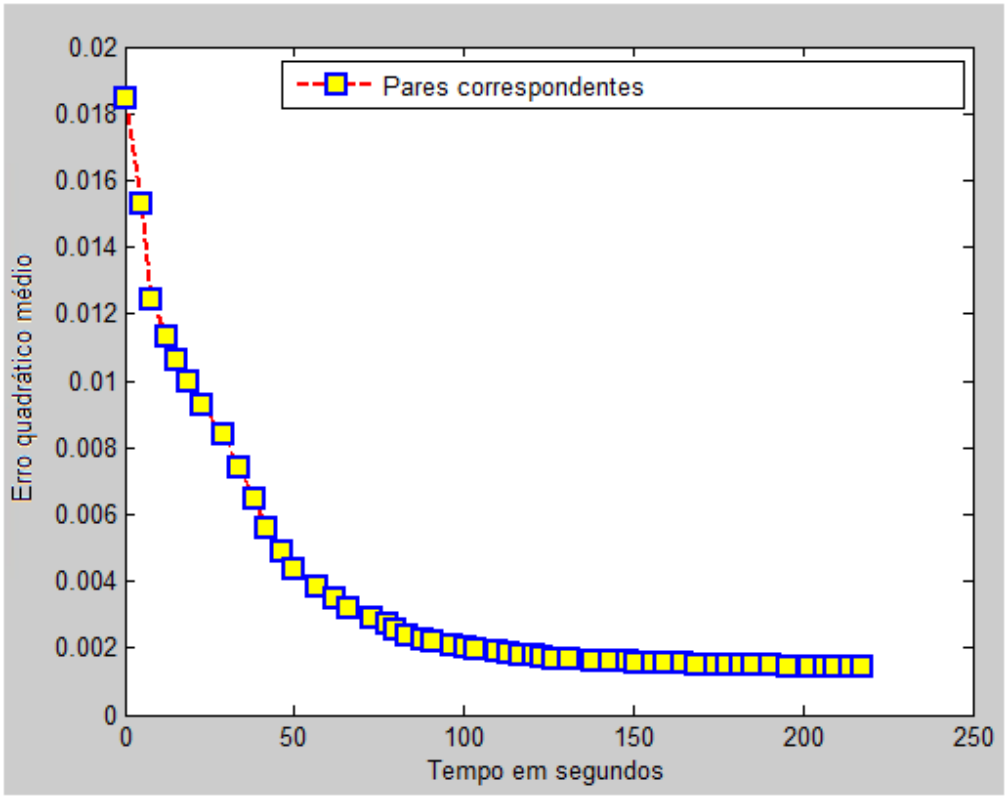


Figura 4.28 Erro quadrático médio obtido utilizando o algoritmo *ICP* "força bruta" com 50 iterações.

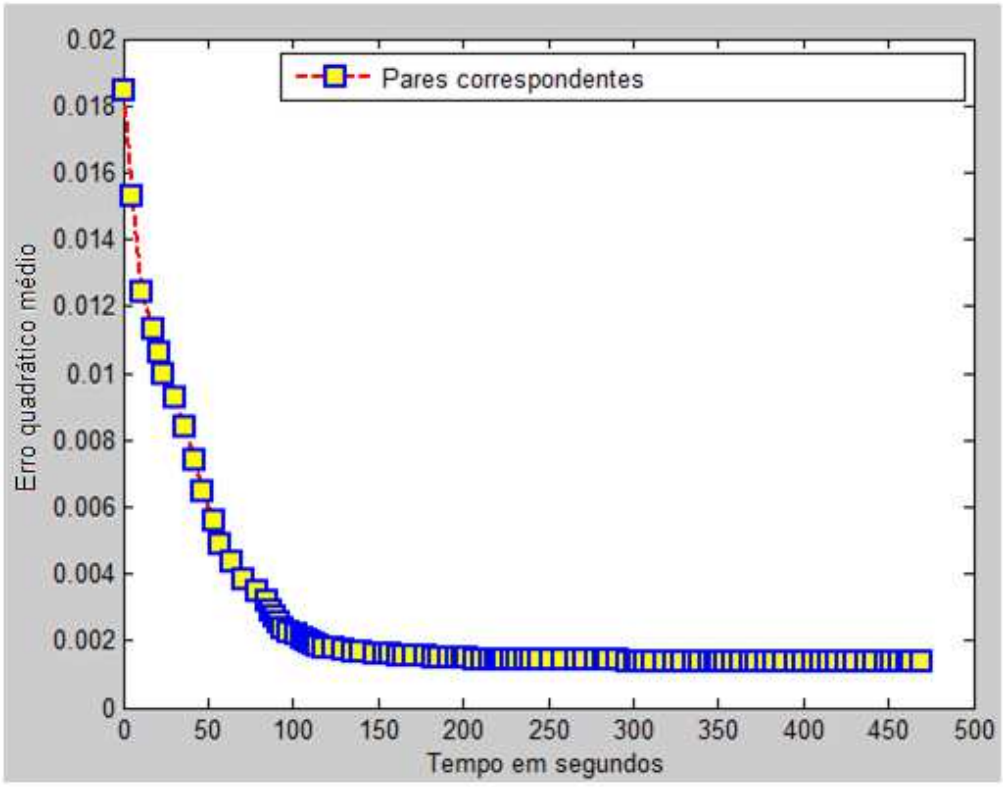


Figura 4.29 Erro quadrático médio obtido utilizando o algoritmo *ICP* "força bruta" com 100 iterações.

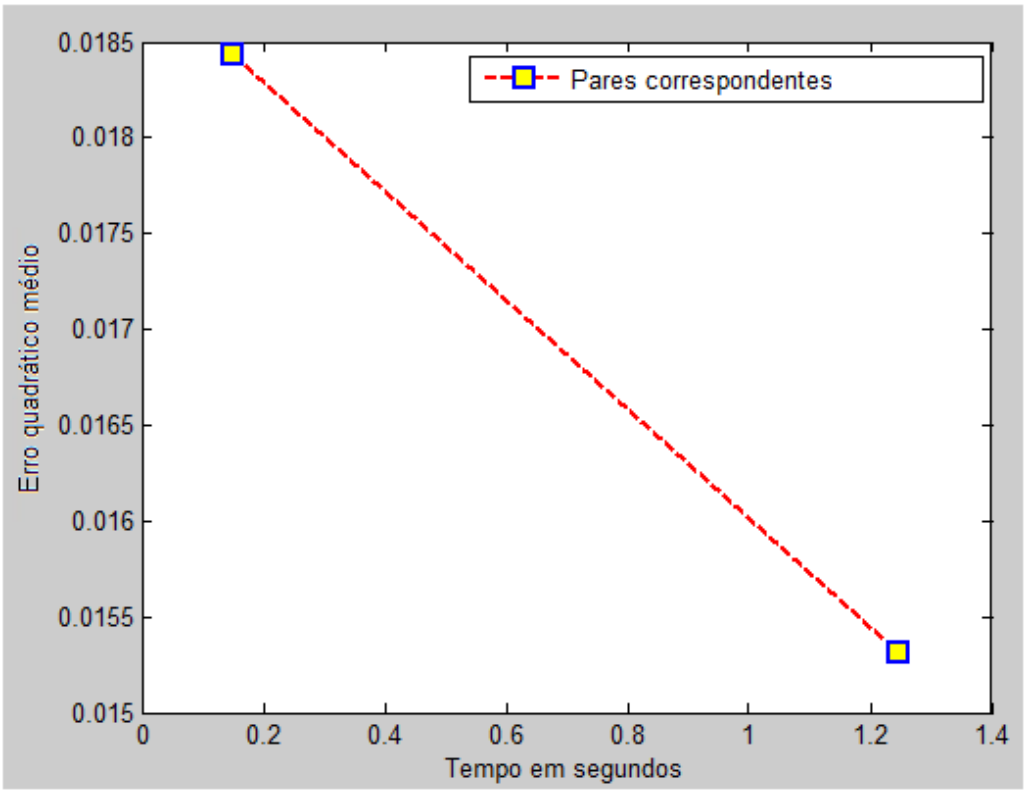


Figura 4.30 Erro quadrático médio obtido utilizando o algoritmo *ICP* "*kd tree*" com 1 iteração.

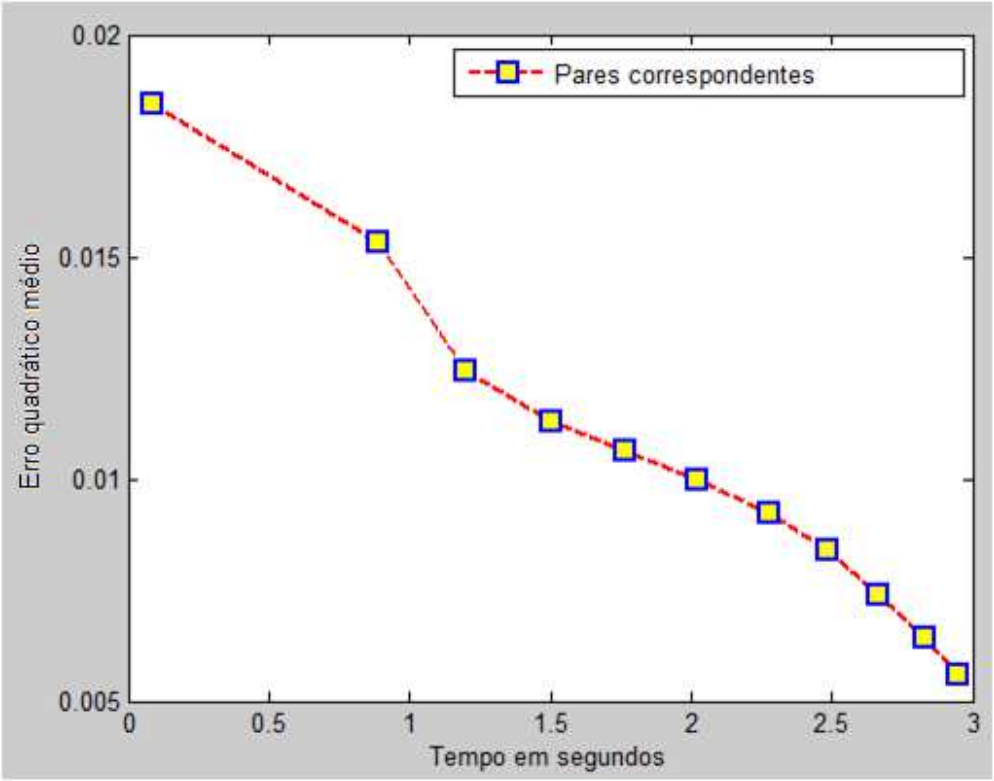


Figura 4.31 Erro quadrático médio obtido utilizando o algoritmo *ICP* "*kd tree*" com 10 iterações.

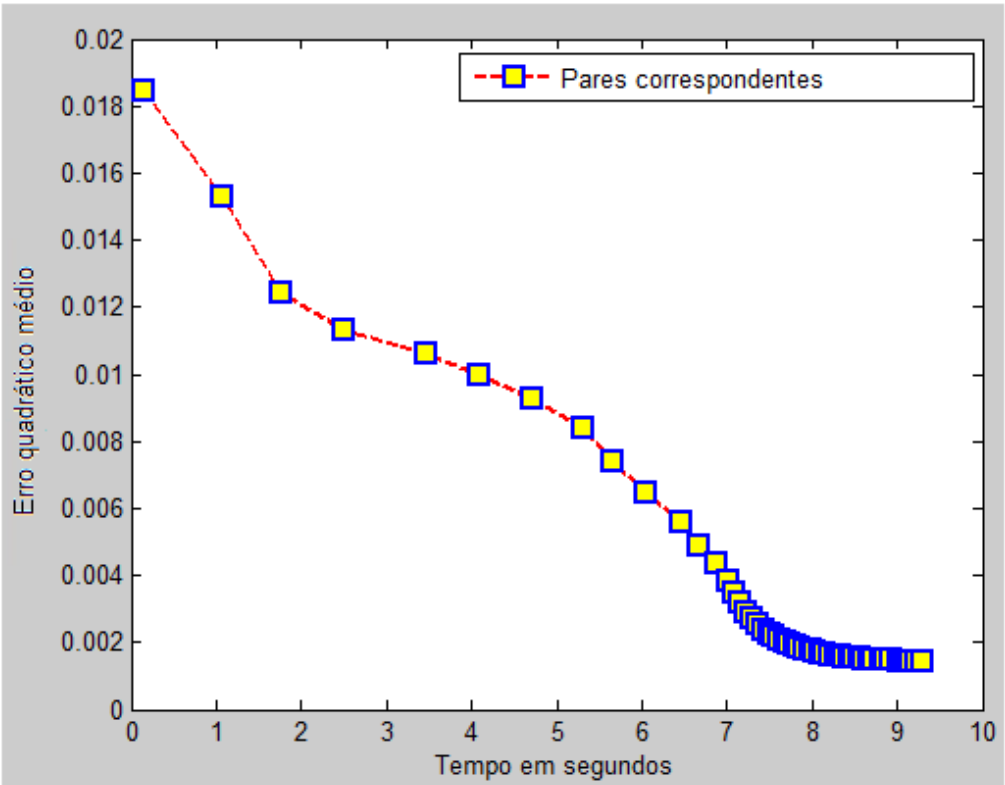


Figura 4.32 Erro quadrático médio obtido utilizando o algoritmo *ICP* "*kd tree*" com 50 iterações.

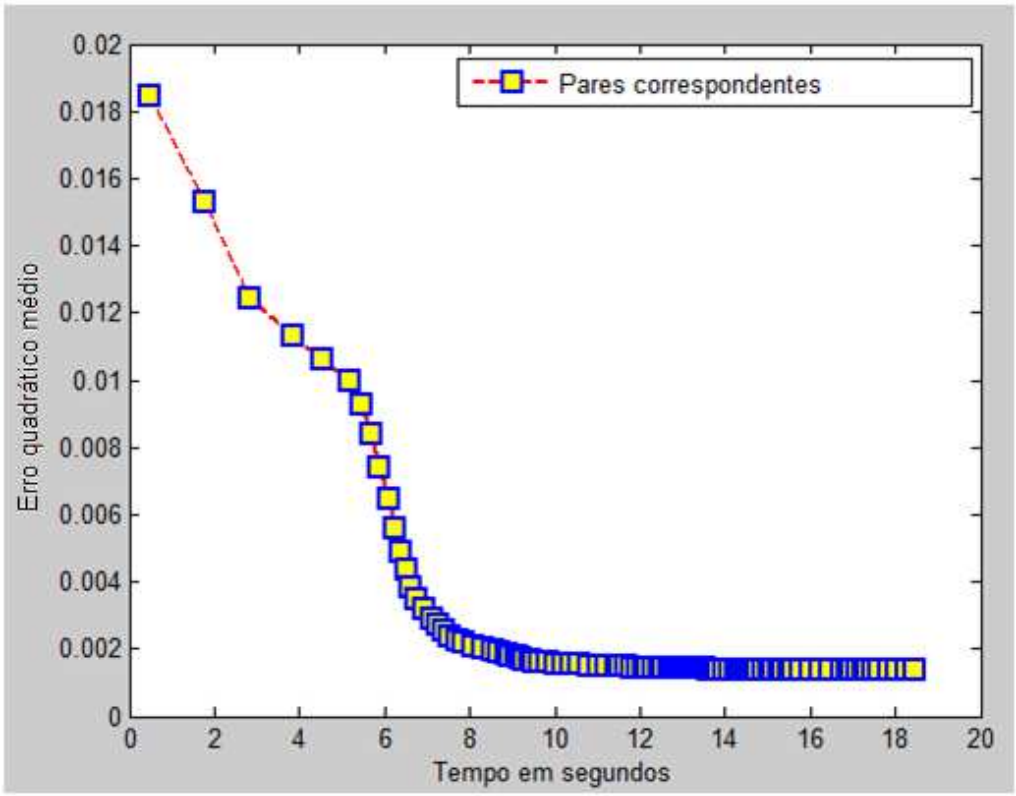


Figura 4.33 Erro quadrático médio obtido utilizando o algoritmo *ICP* "*kd tree*" com 100 iterações.

É possível verificar que o tempo de convergência utilizando o algoritmo "*kd tree*" é muito menor que o tempo utilizado pelo algoritmo "força bruta". A tabela 5.1 mostra a diferença de tempos de convergência entre os dois tipos de algoritmos implementados para diferentes números de iterações.

Tabela 4-1 Diferença de tempos de convergência entre o algoritmo "força bruta" e "*kd tree*" para diferentes números de iterações.

Tipo de algoritmo	Tempo de convergência (em segundos) com 1 iteração	Tempo de convergência (em segundos) com 10 iterações	Tempo de convergência (em segundos) com 50 iterações	Tempo de convergência (em segundos) com 100 iterações
ICP "força bruta"	3.8	21.7	216.1	470.5
ICP "*kd tree*"	1.2	2.9	9.3	18.4

5. APLICAÇÃO E ANÁLISE DOS RESULTADOS

5.1 DESCRIÇÃO DO PROCEDIMENTO E IMAGENS UTILIZADAS

Para testar os algoritmos construídos foi realizada uma reconstrução de objetos aproveitando as propriedades do algoritmo *ICP*. No experimento desenvolvido foi utilizada uma estrutura que permitisse deixar o *scanner* (kinetic®) em uma posição fixa e uma base móvel que permitisse movimentar o objeto de maneira estável. Como estrutura fixa para o digitalizador foi utilizada uma mesa óptica alta e como base móvel foi utilizada uma cadeira, como amostra a Figura 5.1.

Figura 5.1 Imagem do ambiente implementado para o modelamento de objetos, mesa e cadeira.

O objetivo do modelamento foi a digitalização do rosto e cabeça de uma pessoa, com a finalidade de reconstruí-la dentro do computador por meio da renderização de nuvens de pontos. A ideia foi colocar a pessoa (a ser digitalizada) na cadeira móvel e começar a tomar as imagens desde cada um dos ângulos de interesse para o projeto.

Deixando o digitalizador apontando em uma só direção, foram tomadas imagens de vários ângulos conhecidos de um objeto, armazenando os ângulos a partir dos quais foram escaneadas as imagens, para posteriormente determinar uma transformação inicial do sistema para as diferentes posições das vistas. O alinhamento foi alcançado com a manipulação dos parâmetros extrínsecos (rotação e translação), para obter a reconstrução do objeto de forma mais precisa, utilizando oito vistas de diferentes posições (ver Figura

5.2).

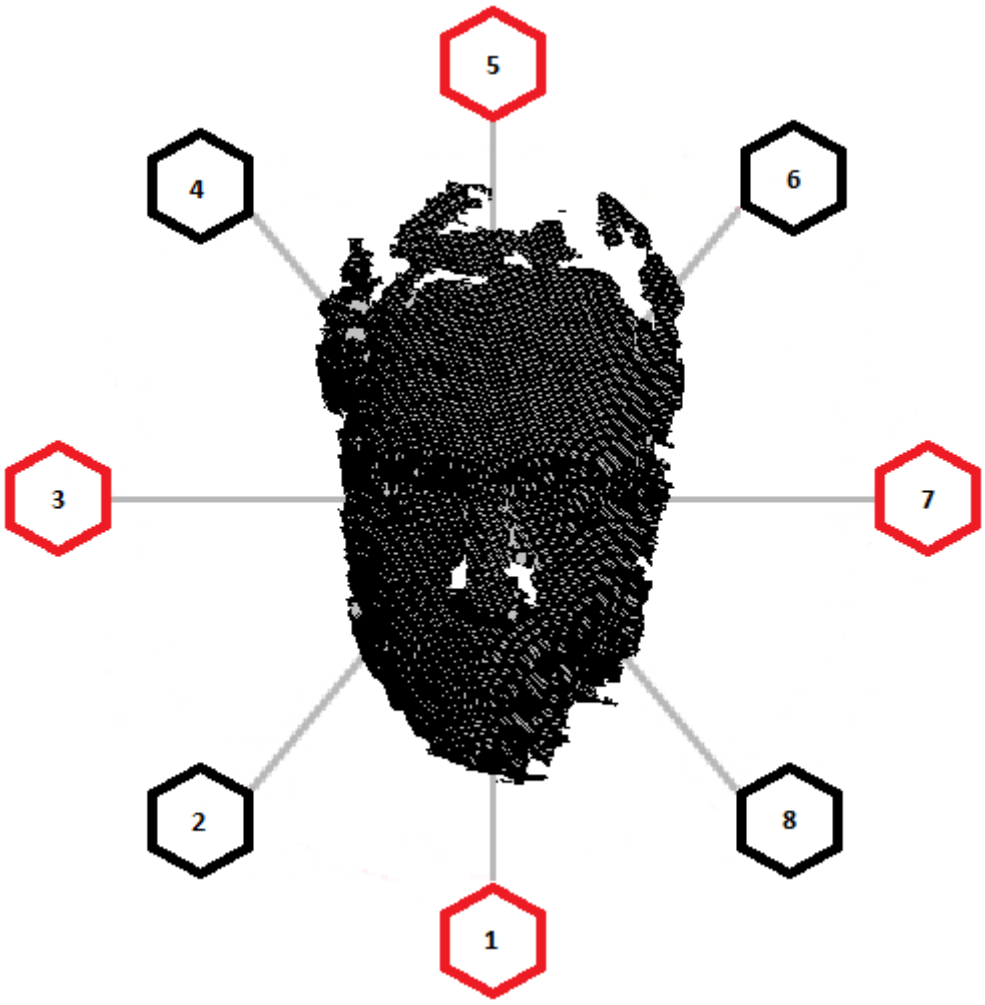


Figura 5.2 Posição das vistas obtidas com a câmera (digitalizador) ao redor do objeto.

Na Figura 5.2, as vistas nº 1, 3, 5 e 7 (vermelhas) foram tomadas nos respectivos ângulos 0º, 90º, 180º e 270º graus com o digitalizador. As outras vistas, nº 2, 4, 6, 8 (pretas), foram tomadas em ângulos não calibrados (desconhecidos). A ideia principal é realizar o registro das nuvens de pontos entre as vistas 1 e 2, 3 e 4, 5 e 6, 7 e 8 para, posteriormente, utilizando o alinhamento inicial das vistas 1, 3, 5, e 7, realizar o modelamento do objeto. As imagens registradas são mostradas nas Figuras 5.3 até a Figura 5.10, para cada uma das vistas.

Figura 5.3 Vista frontal do rosto, correspondente à “vista 1”.

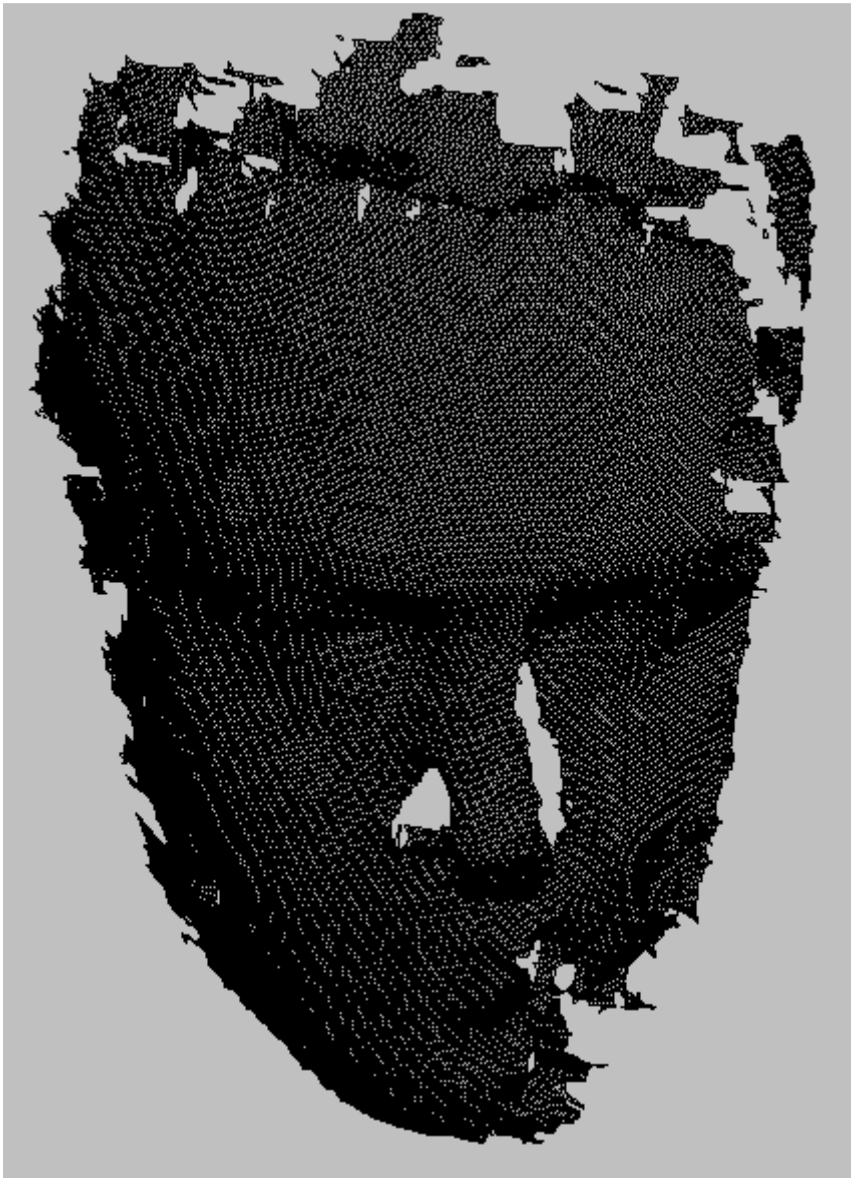


Figura 5.4 Vista frontal do rosto, correspondente à “vista 2”.

Figura 5.5 Vista lateral direita do rosto, correspondente à “vista 3”.

Figura 5.6 Vista lateral direita do rosto, correspondente à “vista 4”.

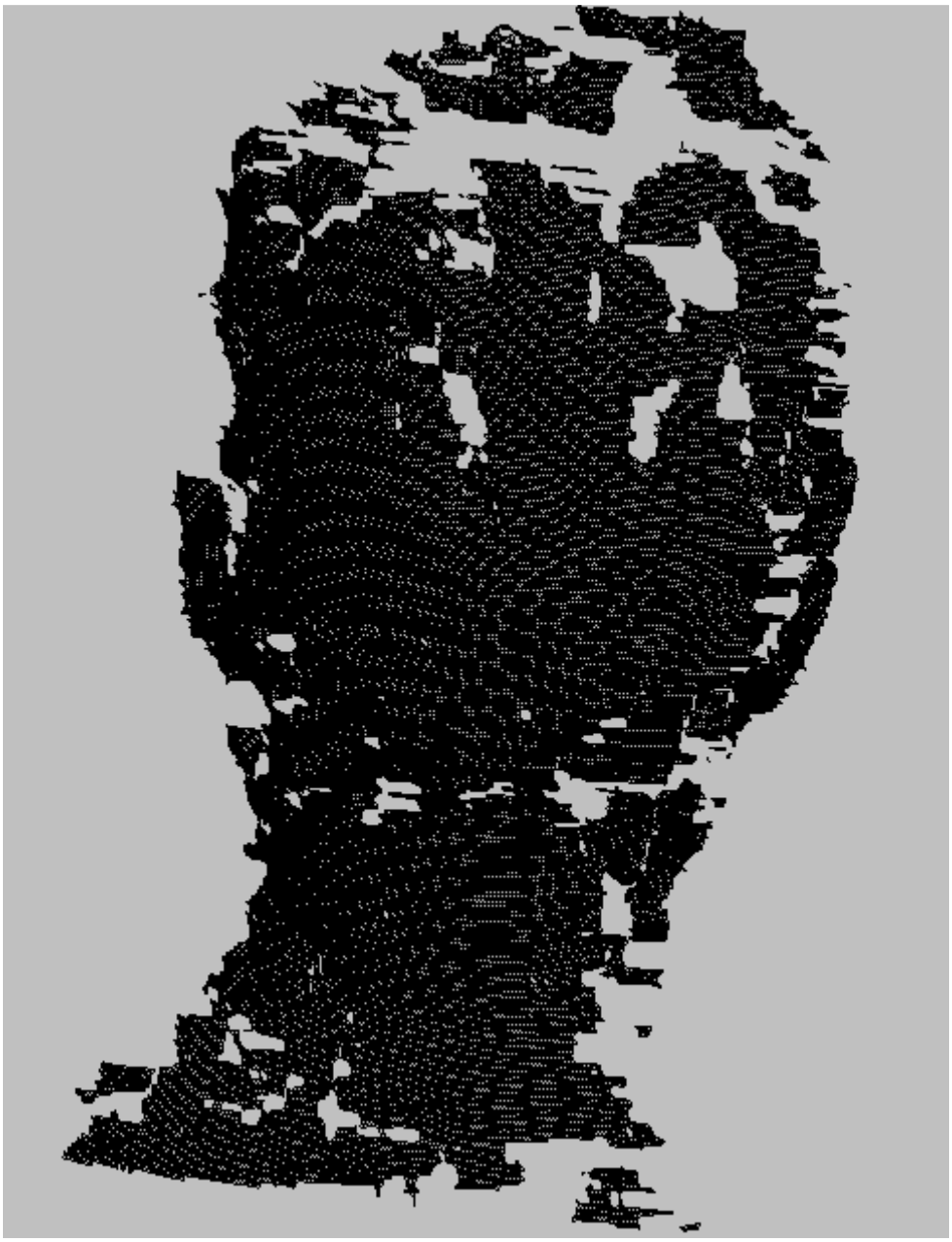


Figura 5.7 Vista posterior do rosto, correspondente à “vista 5”.

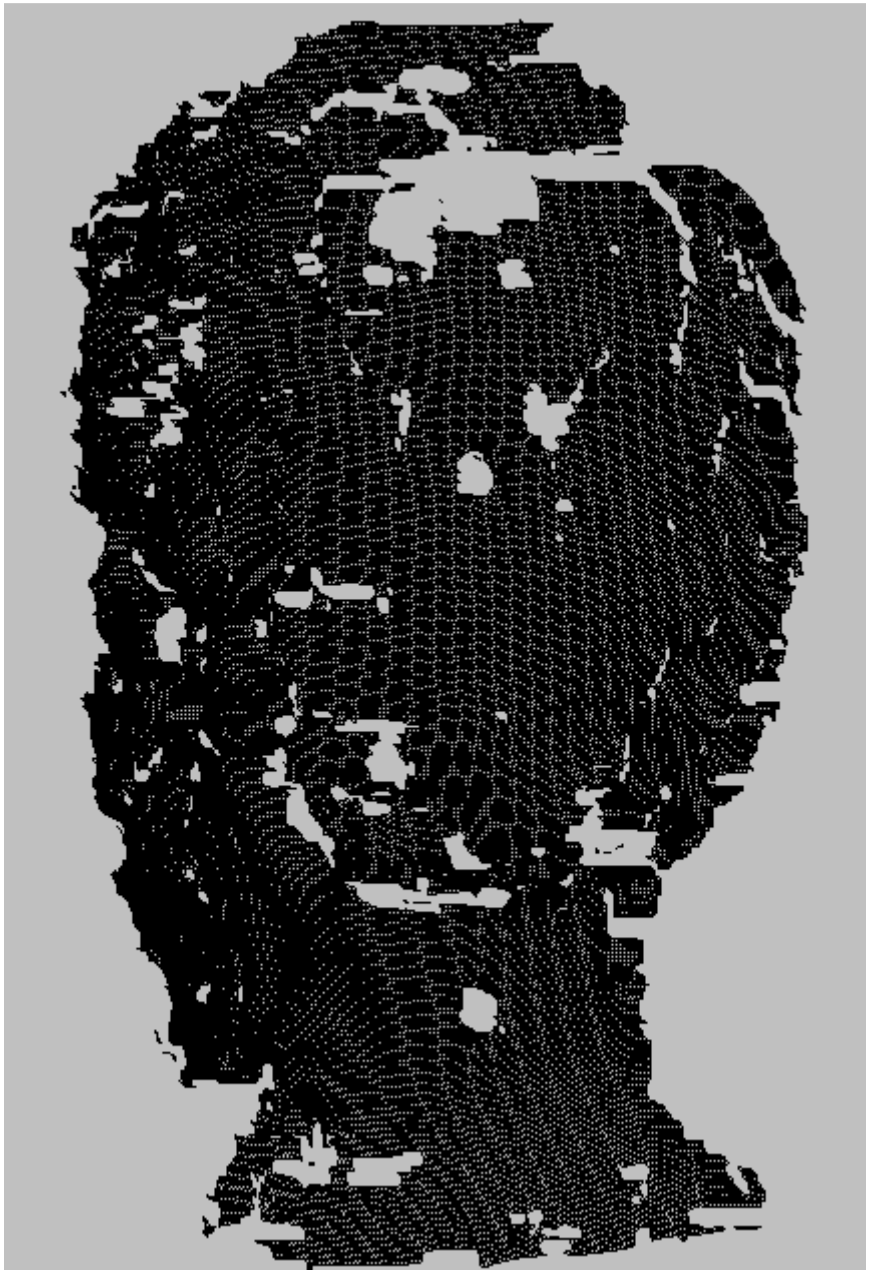


Figura 5.8 Vista posterior do rosto, correspondente à “vista 6”.

Figura 5.9 Vista lateral esquerda do rosto, correspondente à “vista 7”.

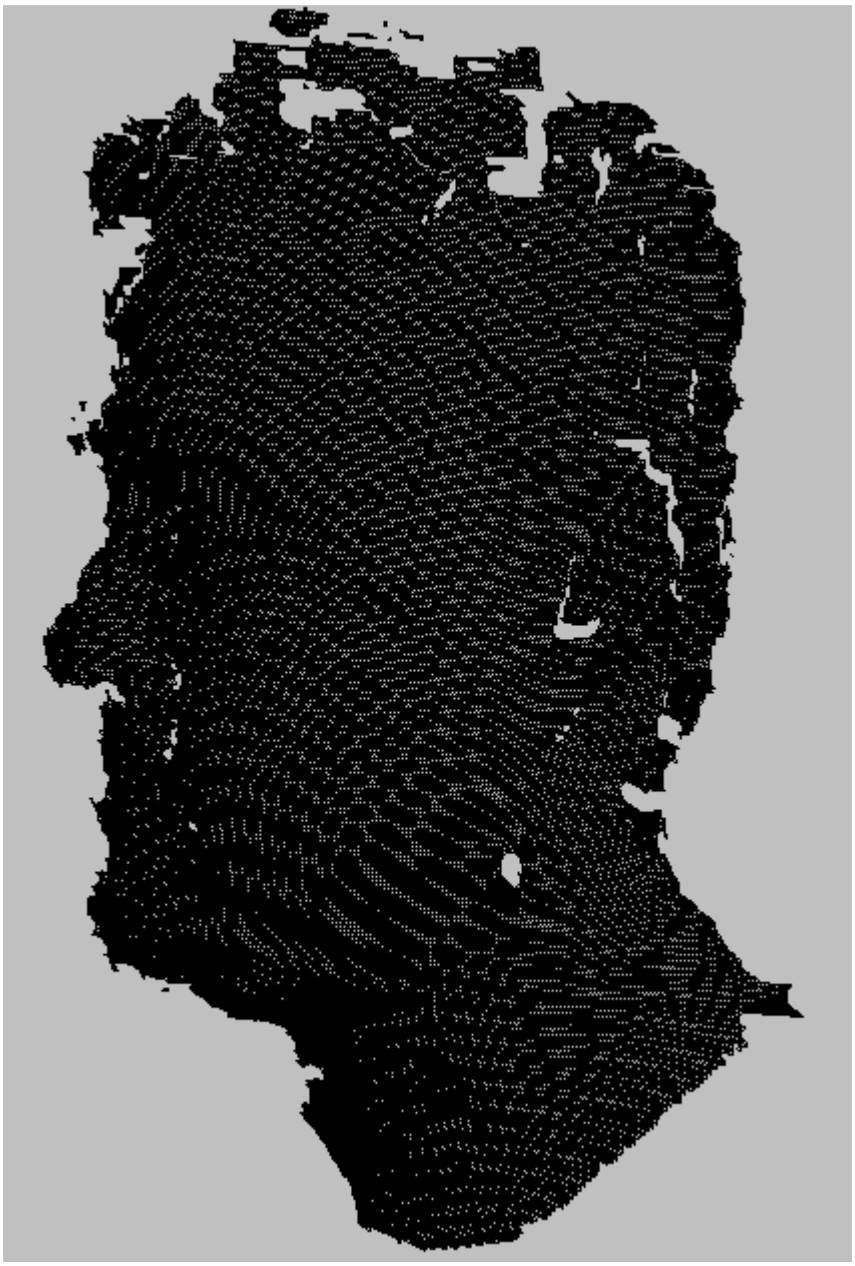


Figura 5.10 Vista lateral do rosto, correspondente à “vista 8”.

O uso do algoritmo *ICP* para realizar o registro das nuvens de pontos em pares permite tirar os “*Dropouts*” das imagens. A existência de “*Dropouts*” dentro das imagens registradas é devido à resolução do “*Kinetic*®”, que não é comparável com a resolução de outros tipos de digitalizadores comerciais, fazendo com que certas partes da superfície dos objetos não sejam digitalizadas.

5.2 GRÁFICOS OBTIDOS

A modelagem obtida é mostrada nas Figuras 5.11 e 5.12 com dois diferentes números de iterações (10 e 100) utilizando o algoritmo ICP "*kd tree*".

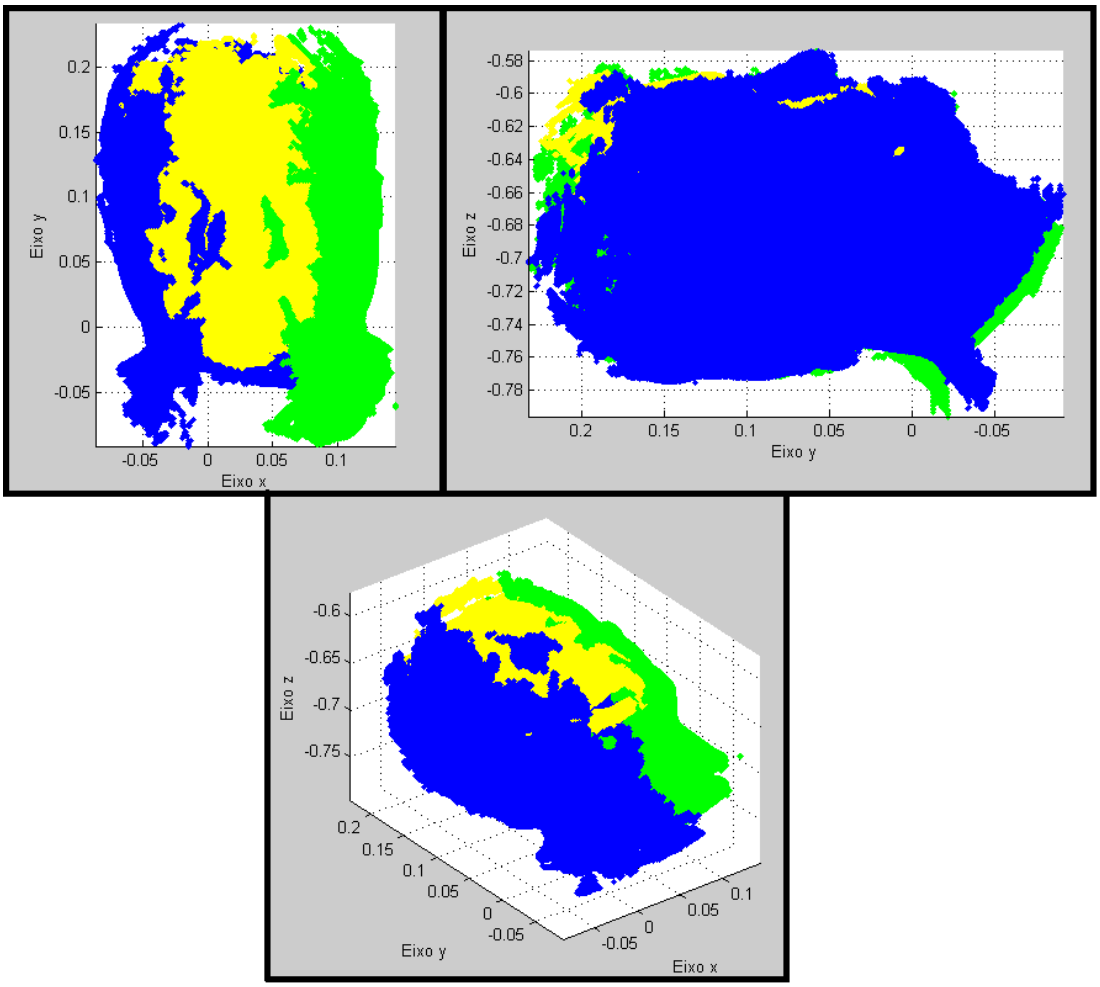


Figura 5.11 Modelamento do objeto com o algoritmo *ICP* "*kd tree*" com 10 iterações. As cores representam as vistas laterais, frontal e posterior do rosto.

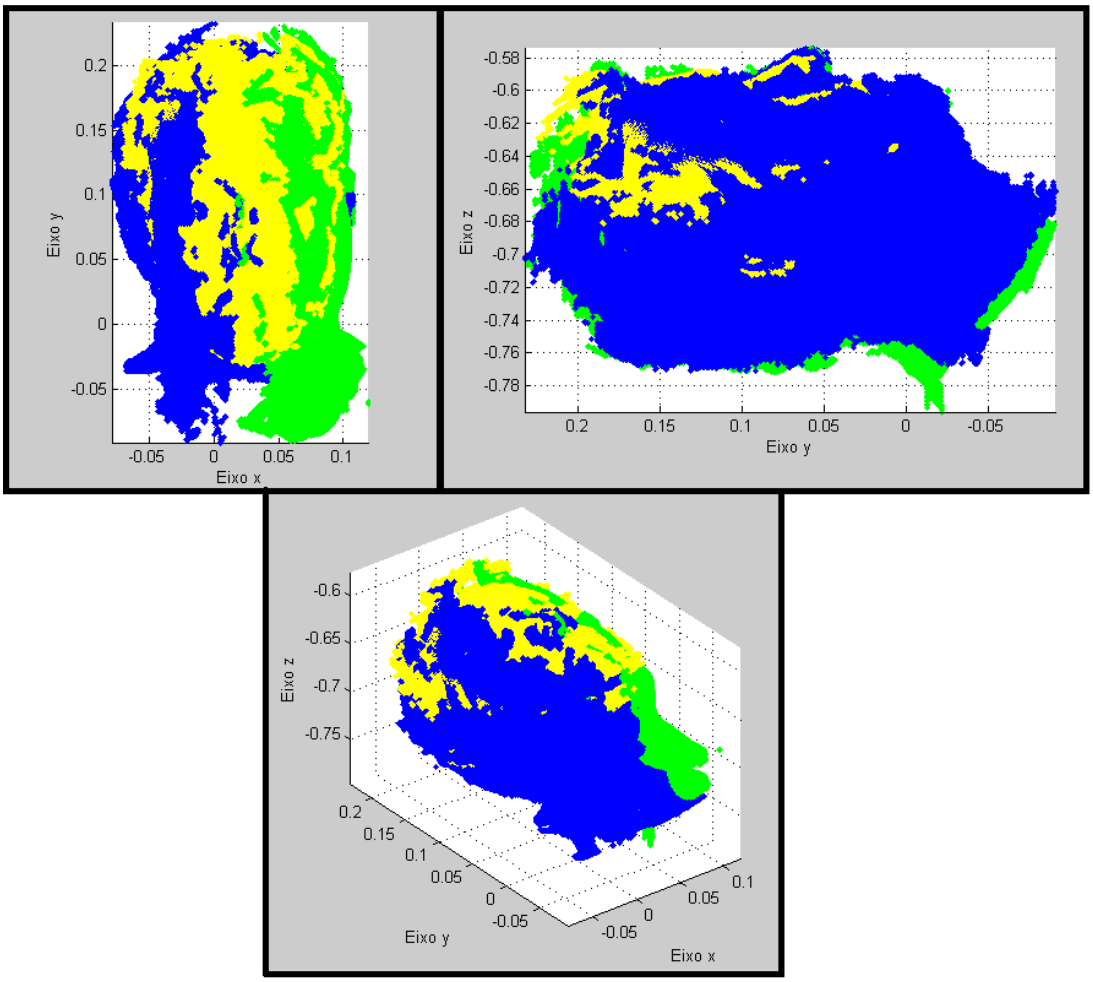


Figura 5.12 Modelamento do objeto com o algoritmo *ICP* "*kd tree*"com 100 iterações. As cores representam as vistas laterais, frontal e posterior do rosto.

Embora o modelamento não fosse perfeito, é possível verificar que, ao utilizar mais iterações do algoritmo, o alinhamento entre as imagens vai melhorando ao fazer um melhor registro das nuvens de pontos, apagando certos defeitos (*Dropouts*) registrados pelo digitalizador (ver Figura 5.13).

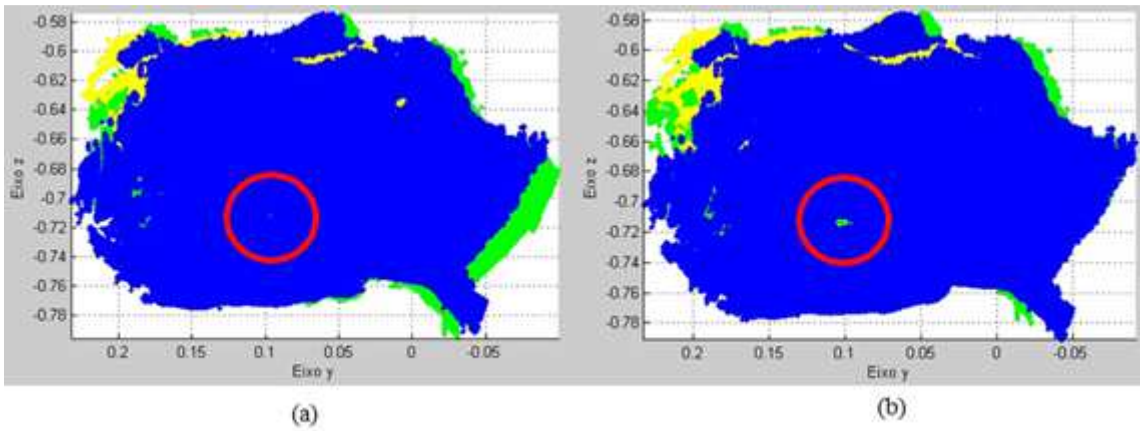


Figura 5.13 Melhora no modelamento utilizando mais iterações dentro do algoritmo *ICP* (a) 10 iterações. (b)1 iteração.

5.3 RESULTADOS DO ERRO QUADRÁTICO MÉDIO NO REGISTRO DAS IMAGENS NO MODELAMENTO DE OBJETOS

Em cada um dos registros feitos sobre as nuvens de pontos no processo de modelamento, foram obtidos os gráficos do Erro Quadrático Médio com respeito ao tempo, como mostram as Figuras 5.15, 5.16, 5.17 e 5.18.

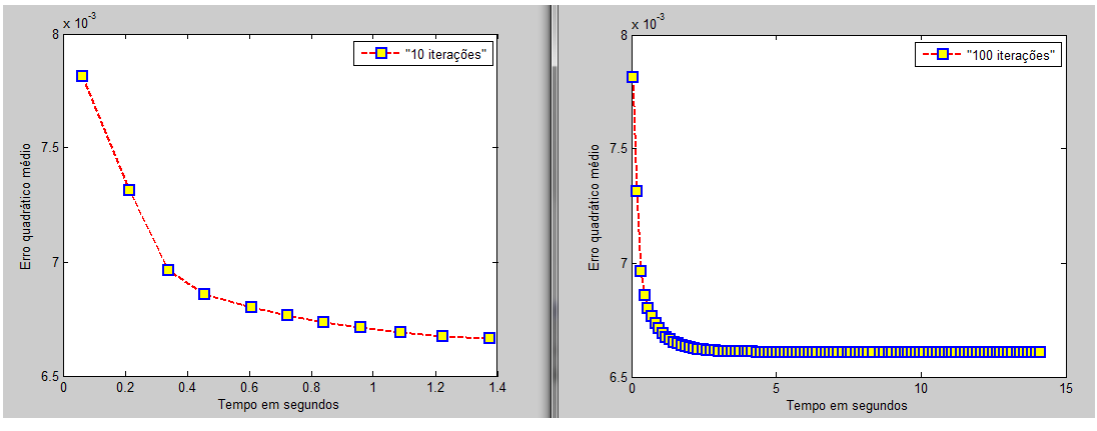
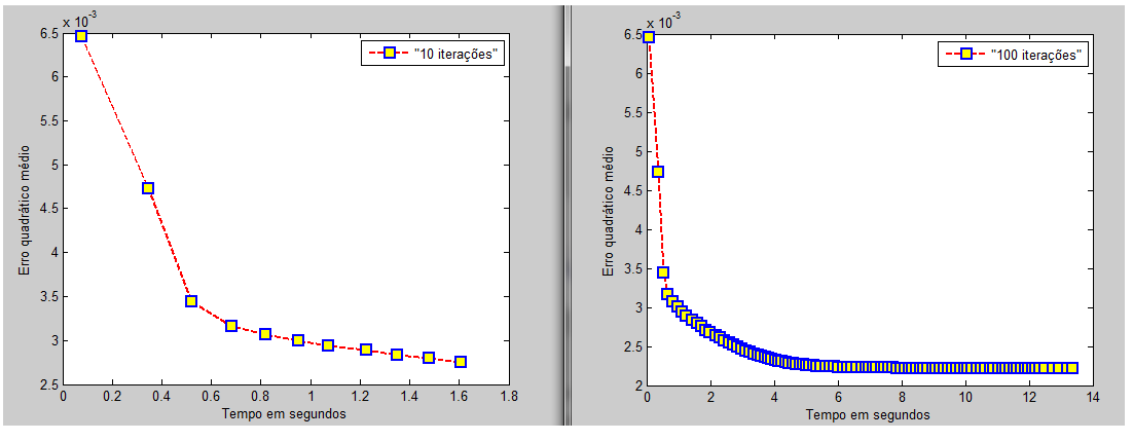
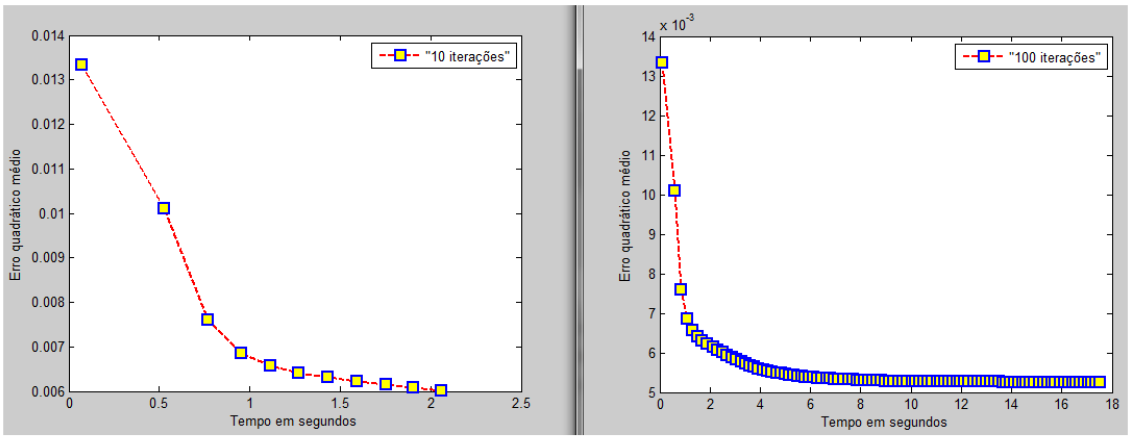


Figura 5.14 (a) “Erro Quadrático Médio” no registro das vistas 1 e 2, utilizando 10 iterações. (b)“Erro Quadrático Médio” no registro das vistas 1 e 2, utilizando 100 iterações.



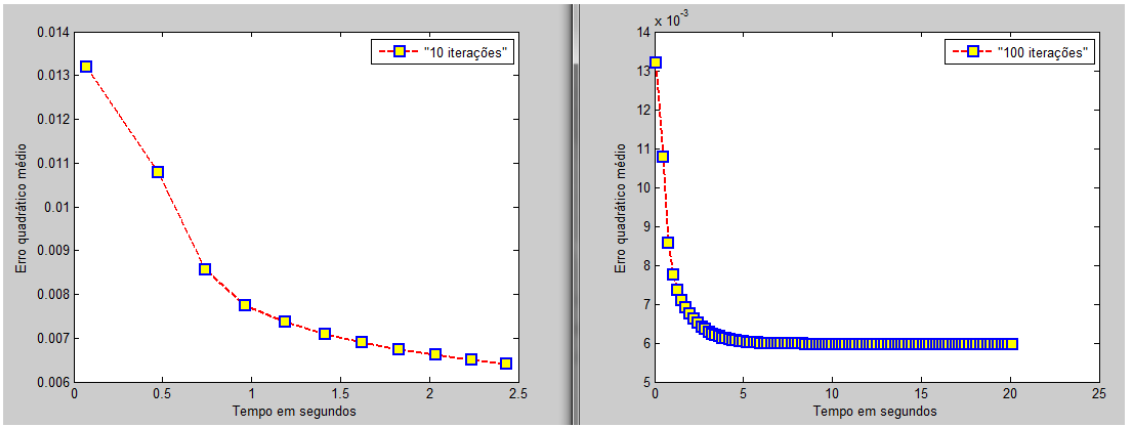
(a) **(b)**

Figura 5.15 (a) “Erro Quadrático Médio” no registro das vistas 3 e 4, utilizando 10 iterações. (b) “Erro Quadrático Médio” no registro das vistas 3 e 4, utilizando 100 iterações.



(a) **(b)**

Figura 5.16 (a) “Erro Quadrático Médio” no registro das vistas 5 e 6, utilizando 10 iterações. (b) “Erro Quadrático Médio” no registro das vistas 5 e 6, utilizando 100 iterações.



(a) **(b)**

Figura 5.17 (a) “Erro Quadrático Médio” no registro das vistas 7 e 8, utilizando 10 iterações. (b) “Erro Quadrático Médio” no registro das vistas 7 e 8, utilizando 100 iterações.

5.4 ALGORITMO “MODELAMENTO DE OBJETOS”

OBJETIVO

Tendo um conjunto de imagens de diversas vistas de um objeto, faz o modelamento do objeto utilizando as nuvens de pontos alinhadas com o uso do algoritmo *ICP*.

1- Carregar um par de nuvens de pontos (vista 1 e 2 das Figuras 5.3 e 5.4.) entre em qualquer um dos ângulos 0º e 90º, 0º≤ângulo<90º;

2- Fazer o registro das duas nuvens de pontos utilizando o algoritmo *ICP* e aplicar uma transformação rígida com relação à primeira nuvem de pontos;

3- Carregar outro par de nuvens de pontos (vista 3 e 4 das Figuras 5.5 e 5.6) entre qualquer um dos ângulos 90º e 180º, 90º≤ângulo<180º;

4- Repetir o passo número “2” com o novo par de nuvens de pontos;

5- Carregar outro par de nuvens de pontos (vista 5 e 6 das Figuras 5.7 e 5.8) entre qualquer um dos ângulos 180º e 270º, 180º≤ângulo<270º;

6- Repetir o passo número “2” com o novo par de nuvens de pontos;

7- Carregar outro par de nuvens de pontos (vista 7 e 8 das Figuras 5.9 e 5.10) entre qualquer um dos ângulos 270º e 360º, 270º≤ângulo<360º;

8- Desenhar e armazenar todas as nuvens de pontos transformadas.

6. DISCUSSÃO E CONCLUSÕES

O conhecimento do ambiente de estudo é fundamental para a construção de modelos digitais de objetos. Um bom alinhamento inicial depende de quanto se conhece o sistema, ou seja, quais são os valores de rotação e translação que uma das imagens precisa para garantir bom registro. Embora o alinhamento seja fundamental para o modelamento, é possível utilizar técnicas mais avançadas para que este alinhamento possa ser feito sem intervenção humana.

A convergência do algoritmo *ICP* é um parâmetro qualitativo, já que não existe um método teórico para provar sua convergência. Não obstante, o parâmetro utilizado neste trabalho, o "Erro Quadrático Médio", deu um indicativo da velocidade na qual as distâncias entre os pontos correspondentes diminuem entre os dois algoritmos do *ICP* implementados, sendo evidente, ao observar os gráficos dos resultados, que as velocidades de convergência do algoritmo *ICP "kd tree"* foram maiores, além de mostrar que a intervenção do ruído gaussiano não atrapalha o processo de registro utilizado nos dois tipos de algoritmos do *ICP*.

O uso do programa *Matlab* deu a vantagem no manejo de matrizes na programação do algoritmo, mas introduz uma desvantagem no tempo de execução das interfaces do sistema, já que os cálculos dos processos que utiliza o algoritmo não são feitos em tempo real. Mas essa não foi uma desvantagem neste trabalho, já que o tempo de execução não era um fator determinante para alcançar os objetivos do trabalho.

O algoritmo *ICP* com a melhora do algoritmo de busca "*kd tree*" converge mais rápido que o algoritmo *ICP* convencional. Isso é devido à economia de tempo ao não realizar todos os cálculos para encontrar a distância mais curta entre pontos correspondentes, como é feito no "força bruta". A Tabela 4.1 mostra que o algoritmo "*kd tree*" converge mais rapidamente que o algoritmo "força bruta".

A técnica de modelamento implementada neste trabalho oferece a vantagem de reconstruir um objeto estático com poucas imagens, mas não permite realizar um modelamento de um sistema dinâmico. A grande vantagem do projeto foi automatizar o problema de registro e modelamento de objetos, tirando grande parte da intervenção humana ao aplicar algoritmos robustos.

7. TRABALHOS FUTUROS

Muitas melhoras podem ser feitas dentro do algoritmo *ICP* e dentro do sistema implementado que foi realizado neste trabalho. As melhorias podem ser feitas em diferentes partes. As propostas para ser realizadas em trabalhos futuros são as seguintes:

- Iluminação: algumas condições dentro do ambiente podem interferir de maneira direta sobre os resultados de projetos que trabalhem com dispositivos como câmeras. Devido a esse problema, seria necessário realizar um estudo rigoroso de como é feita o tipo de iluminação sobre os objetos que vão ser registrados com o digitalizador a laser, para melhorar o registro das nuvens de pontos.

- Tempo real: a utilização de programas e sistemas operacionais que trabalhem em tempo real como *JAVA* e *LINUX* poderia trazer um avanço ao trabalhar com tamanhos de nuvens de pontos maiores, já que se aumentarmos o número de pontos a serem analisados, o custo computacional seria maior, traduzindo-se em um ganho de tempo, além de ter a possibilidade de se fazer um estudo com sistemas não estáticos que precisassem de respostas em tempo real.

- Remoção de "*outliers*": um dos problemas do registro é a remoção dos "*outliers*". Poderia ser feita a adição de um algoritmo como o *RANSAC* que tire os "*outliers*". Além disso, os objetos que não sejam de interesse para o estudo (como a parede de fundo) fossem tirados, oferecendo um registro automático sem precisar da ajuda humana e de outros programas como o "*Meshlab*" para tirá-los.

- Conhecimento do sistema: o fato de o algoritmo precisar dos parâmetros extrínsecos para realizar o modelamento de objetos faz com que a intervenção humana seja altamente necessária. O uso de um robô poderia realizar a função do operador ao fazer o alinhamento, se fosse feito um algoritmo que permitisse identificar a localização do objeto

dentro do sistema e traduzir essa posição em parâmetros extrínsecos.

Essas melhoras poderiam ser implementadas especificamente para realizar um projeto focado em fazer um modelamento de objetos totalmente automatizado, com digitalizações de qualidade, tirando qualquer intervenção humana.

REFERÊNCIAS BIBLIOGRAFICAS

Arun, K., Huang, T., and Blostein, S., (1987).Least-Squares Fitting of Two 3-D Point Sets. Pattern Analysis And Machine Intelligence, PAMI, Vol. 9, No. 5, pp. 698-700.

Bae, K., D., Lichti, D. D., (2005). A Method For Automated Registration of Unorganised Point Clouds. ISPRS Journal of Photogrammetry & 63, pp. 36-54.

Blais, G., Levine, M. D., (1995). Registering Multiview Range Data to Create 3D Computer Objects.Pattern Analysis And Machine Intelligence, Vol. 17, Issue 8, pp. 820-824.

Bentley, J. L., (1980). Multidimensional Divide and Conquer. Communications of the ACM, pp. 214-229.

Besl, P. J. e McKay, N. D. (1992). A Method for Registration of 3-D Shapes.Pattern Analysis and Machine Intelligence, PAMI, Vol. 14, No. 2, pp. 239-256.

Birn, M., Holtgrewe, M., Sanders, P., Singler, J., (2010).Simple and Fast Nearest Neighbor Search, Workshop on Algorithm Engineering and Experiments, pp. 43-54.

Boehnen, C., Flynn, P., (2005). Accuracy of 3D Scanning Technologies in a Face Scanning Scenario.Fifth International Conference on 3-D Digital Imaging and Modeling, pp. 310-317.

Boykov, Y., Kolmogorov, V., (2004). An Experimental Comparison of Min-Cut/Max-Flow

Algorithms for Energy Minimization in Vision, Pattern Analysis And Machine Intelligence, Vol. 26, No. 9, pp. 1124-1137.

Champleboux, G., Lavallee, S., Szeliski, R., Brunie, L., (1992). From Accurate Range Imaging Sensor Calibration To Accurate Model-Based 3-D Object Localization. Conference on Computer Vision and Pattern Recognition, pp. 83-89.

Chen, Y. eMedioni, G. (1992). Object modelling by registration of multiple range images. Image and Vision Computing, Vol. 10, N0. 3, pp. 145-155.

Chen, C., Hung, Y., Cheng, J., (1999). RANSAC-based DARCES: A New Approach To Fast Automatic Registration Of Partially Overlapping Range Images. Pattern Analysis and Machine Intelligence, Vol. 21, No. 11, pp. 1229-1234.

Dorai, C., Weng, J., Jain, A. K., (1994). Optimal Registration Of Multiple Range Views. 12th International Conference on Pattern Recognition, pp. 569-571.

Dorai, C., Weng, J., Jain, A. (1998). Registration and Integration of Multiple Object Views for 3D Model Constrution.PAMI, Vol. 20, No. I .pp. 83-89.

Enqvist, O., Jiang, F., Kahl, F., (2011).A brute-force algorithm for reconstructing a scene from two projections. Conference On Computer Vision And Pattern Recognition, pp. 2961-2968.

Eskola, S. R., 2001, Binary Space Partioning Trees and Polygon Removal in Real Time 3D Rendering, Information Technology Computing Science Department Uppsala University, Suécia: Theses.

Fuchs, H., Abram, G. D., Grant, E. D., (1983). Near Real-Time Shaded Display of Rigid Objects.Computer Graphics, vol. 17, pp. 65-72.

Georgopoulos A., Ioannidis Ch., Valanis A., (2010). Assessing the Performance of a Structured Light Scanner. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXVIII, pp. 250-255.

Godin G., Rioux M., Baribeau R., (1994). Three-dimensional Registration Using Range and Intensity Information, Videometrics III, Vol. 2350, pp. 279-290.

Gomes, J., 2001. LocalisationOf a Mobile RobotUsing Laser Scanner AndReconstructed 3D Models, Universidade tecnica de Lisboa, Departamento de Engenharia Electrotécnica e de Computadores, Portugal: Thesis.

Gu, L.F., (1997). Visual guidance of robot motion. International Conference on Intelligent Processing Systems, Vol. 2, pp. 1242-1246.

Hähnel, D., Thrun S., Burgard, W., (2003). An Extension of the ICP Algorithm for Modeling Nonrigid Objects With Mobile Robots. 18th International Joint Conference on Artificial Intelligence, pp. 915-920.

Hodge, V. J., Austin, J., (2004).A Survey of Outlier Detection Methodologies. Artificial Intelligence Review, Vol. 22, pp. 1-43.

Horn, B.,(1987).Closed-Form Solution Of Absolute Orientation Using Unit Quatemions. JOSA A, Vol. 4, No. 4, pp. 629-642.

Horn, B., Hilden, H., and Negahdaripour, S., (1988).Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices. Journal of the Optical Society of America A: Optics, Image Science, and Vision, Vol. 5, No. 7, pp. 1127-1135.

Hwang, Y., Han, B., Ahn, H., (2012). A Fast Nearest Neighbor Search Algorithm by Nonlinear Embedding. Conference On Computer Vision And Pattern Recognition, pp. 3053-3060.

Jarvis, R. A., (1983). A Perspective on Range Finding Techniques for Computer Vision. Pattern Analysis And Machine Intelligence, PAMI, Vol. 5, Issue 2, pp. 122-139.

Johnson, A., Kang S. B., (1997). Registration and Integration of Textured 3-D Data.International Conference on Recent Advances in 3-D Digital Imaging and Modeling, pp. 234 - 241.

Kakadiaris, I.,Passalis, G., Toderici, G., Murtuza, N., Theoharis, T., (2005). Evaluation Of 3D Face Recognition In The Presence Of Facial Expressions: An Annotated Deformable Model Approach. Conference on Computer Vision And Pattern Recognition, pp. 171.

Lerch, T., MacGillivray, M., Domina, T., (2007). 3D Laser Scanning: A Model of Multidisciplinary Research. Journal of Textile and Apparel, Technology Management, Volume 5, Issue 4, pp. 1-22.

Malhotra, A., Gupta, K., Kant, K., (2011). Laser Triangulation for 3D Profiling of Target. International Journal Of Computer Applications, pp. 47-50.

Menq C., Yau H., Lai G., (1992). Automated Precision Measurement of Surface Profile in

CAD-Directed Inspection.IEEE Transactions on Robotics and Automation, Vol 8, No 2, pp. 268-278.

Masuda, T., Sakaue, K., and Yokoya, N., (1996).Registration and Integration of Multiple Range Images for 3-D Model Construction.13th International Conference on Pattern Recognition, Vol. 1, pp. 879-883.

Mitra N., J., Gelfand N., Pottmann H., Guibas L., (2004). Registration of Point Cloud Data from a Geometric Optimization Perspective, Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 22-31.

Naylor B., (1993) Constructing Good Partitioning Trees. Graphics Interface, pp. 1-11.

Nkanza, N., 2005. Image Registration and its Application to Computer Vision: Mosaicing and Independent Motion detection, University of Cape Town, Department of Electrical Engineering, South Africa: Thesis.

Potmesil, M., (1983).Generating Models of Solid Objects by Matching 3D Surface Segments. International Joint Conference on Artificial Intelligence, pp. 1089-1093.

Pulli, K., 1997, Surface Reconstruction and Display from Range and Color Data, University of Washington: Thesis.

Pulli, K., (1999) .Multiview Registration for Large Data Sets. 3DIM, pp. 160-168.

Rusinkiewicz, S., Levoy, M., (2001).Efficient Variants of the ICP Algorithm. Third International Conference On 3D Digital Imaging and Modeling, pp.145-152

Salvi, J., Matabosch, C., Fofi, D., Forest, J., (2007). A Review of Recent Range Image Registration Methods With Accuracy Evaluation. Image and Vision Computing, vol 25, pp. 578-596.

Sandu, L., Topala, F., Bortun, C., Porojan S., (2007).Laser Scanning Applications For Three dimensional Teeth Models Reconstruction. 2nd International Conference On Lasers In Medicine, pp. 271-275.

Silva, L., Bellon, O. R. P., (1995). Robust range image registration using genetic algorithms and the surface interpenetration measure. Series in machine perception and artificial intelligence, Vol. 60.

Simon, D., 1996. Fast and Accurate Shape-Based Registration, Robotics Institute, Carnegie Mellon University: Thesis.

Scheuren, F., Winkler, W. E., (1996). Recursive Merging and Analysis of Administrative Lists and Data. Section on Government Statistics, American Statistical Association, pp. 123-128.

Shlens, J., (2009). A Tutorial on Principal Component Analysis. Center for Neural Science, New York University, pp. 1-12.

Tagare, X. Q., (2006). Overcoming Dropout While Segmenting Cardiac Ultrasound Images. 3rd IEEE International Symposium on Biomedical Imaging, pp. 105-108.

Trucco, E. e Verri, A. (1998).Introductory Techniques for 3-D Computer Vision.Prentice Hall.

Turk, G.,Levoy, M., (1994) Zippered Polygon Meshes from Range Images. Proc. SIGGRAPH, pp. 1-8.

Varady, T., Martin, R. R. e Cox, J. (1997). Reverse Engineering Of Geometric Models - An Introduction.Computer-Aided Design, Vol. 29, No 4, pp. 255-268.

Venâncio, M. (2008). Alinhamento de Imagens Médicas 2D e 3D. Universidade Técnica de Lisboa, Departamento de Engenharia Electrotécnica e de Computadores, Portugal: Thesis.

Weik, S. (1997).Registration of 3-D Partial Surface Models Using Luminance and Depth Information.3DIM, pp. 93-100.

Zhang, Z., (1992). Iterative Point Matching for Registration of Free-Form Curves.International Journal of Computer Vision, Vol. 13, Issue 2, pp. 119-152.

APENDICES

A - *BSP Tree*

Os passos para construir um *BSP* são os seguintes (Fuchs, et al., 1983):

- Calcular a média das coordenadas (em todos os eixos) da nuvem de pontos e identificar o ponto mais próximo como a raiz.

- Identificar os pontos mais próximos da raiz e definir o ramo ao qual pertence cada um dos pontos, utilizando as coordenadas.

- Percorrer o ramo e colocar o ponto como folha no ramo.

Exemplo:

Tendo as seguintes coordenadas de pontos:

- (4,4,4)

- (3,5,2)

- (2,1,3)

- (1,6,2)

- (5,6,8)

- (5,4,2)

- (7,7,1)

Então:

Localizar o nó raiz. Neste caso é o nó (4,4,4) como mostra a Figura A.1.

Figura A. 1 Primeiro nó.

O ponto seguinte é o (3,5,2) que será colocado no ramo esquerdo do nó raiz porque a sua primeira componente (3) é menor que a primeira componente do nó raiz (4), como mostra a Figura A.2.

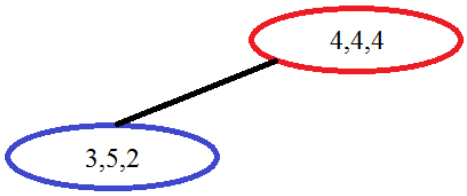


Figura A. 2 Segundo nó.

O ponto seguinte é o (2,1,3) que será colocado no ramo esquerdo do nó raiz porque a sua primeira componente (2) é menor que a primeira componente do nó raiz (4). Nesse nível, o ponto será comparado com o seguinte nó (3,5,2) utilizando a segunda componente. O ponto será colocado no ramo esquerdo do nó (3,5,2) porque a sua segunda componente (1) é menor que a segunda componente (5) do nó (3,5,2), como mostra a Figura A.3

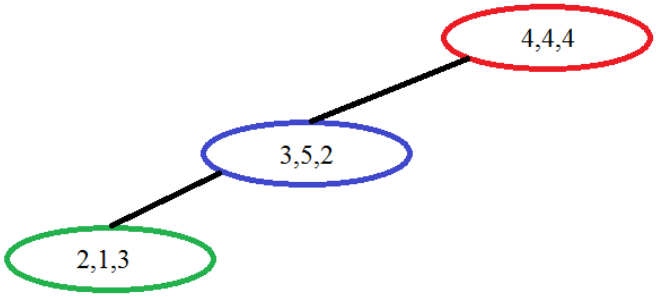


Figura A. 3 Terceiro nó.

O ponto seguinte é o (1,6,2) que será colocado no ramo esquerdo do nó raiz porque a sua primeira componente (1) é menor que a primeira componente do nó raiz (4). Nesse nível, o ponto será comparado com o seguinte nó (3,5,2) utilizando a segunda componente. O ponto será colocado no ramo direito do nó (3,5,2) porque a sua segunda componente (6) é maior que a segunda componente (5) do nó (3,5,2), como mostra a Figura A.4.

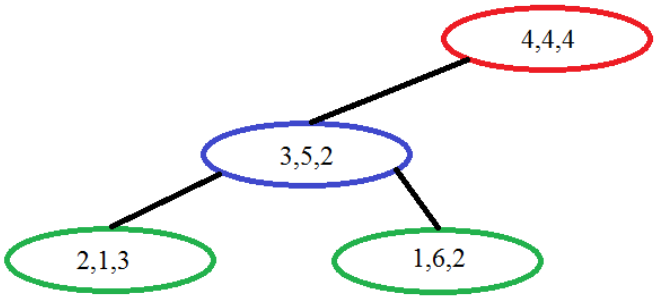


Figura A. 4 Quarto nó.

O ponto seguinte é o (5,6,8) que será colocado no ramo direito do nó raiz porque a sua primeira componente (5) é maior que a primeira componente do nó raiz (4), como mostra a Figura A.5.

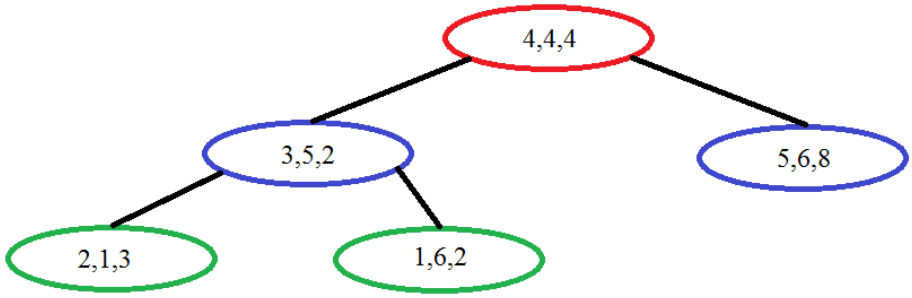


Figura A. 5 Quinto nó.

O ponto seguinte é o (5,4,2) que será colocado no ramo direito do nó raiz porque a sua primeira componente (5) é maior que a primeira componente do nó raiz (4). Nesse nível, o ponto será comparado com o seguinte nó (5,6,8) utilizando a segunda componente. O ponto será colocado no ramo esquerdo do nó (5,6,8) porque a sua segunda componente (4) é menor que a segunda componente (6) do nó (5,6,8), como mostra a Figura A.6.

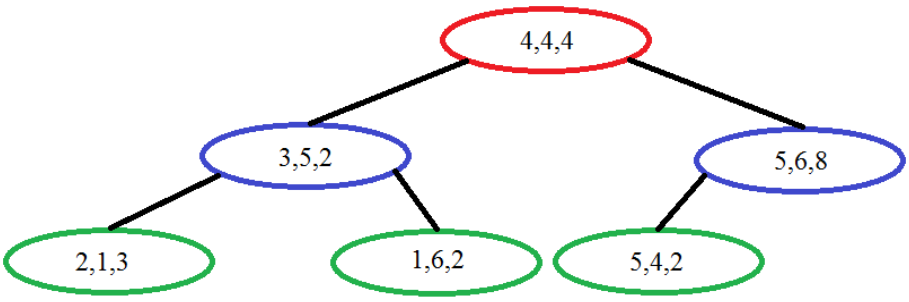


Figura A. 6 Sexto nó.

O ponto seguinte é o (7,7,1) que será colocado no ramo direito do nó raiz porque a sua primeira componente (5) é maior que a primeira componente do nó raiz (4). Nesse nível, o ponto será comparado com o seguinte nó (5,6,8) utilizando a segunda componente. O ponto será colocado no ramo esquerdo do nó (5,6,8) porque a sua segunda componente (4) é menor que a segunda componente (6) do nó (5,6,8), como mostra a Figura A.7.

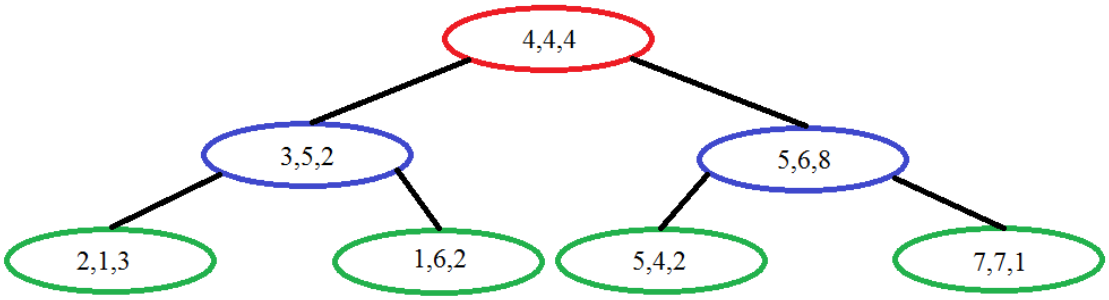


Figura A. 7 Sétimo nó.

B -*Outliers*

Os "*outliers*" são observações que apresentam um grande afastamento dentro das observações normais registradas de um sistema (Hodge e Austin, 2004). A detecção de "*outliers*" é um processo para detectar e remover observações anômalas dentro dos dados de um determinado sistema. Dependendo da área, os "*outliers*" surgem devido a falhas mecânicas, mudanças no comportamento do sistema, erros humanos ou instrumentais entre outros.

A detecção de *outliers* pode identificar falhas no sistema antes que eles afetem as características do sistema em grandes proporções.

A ideia é identificar e remover os erros com efeitos contaminantes dentro da amostra ou conjunto de estudo para purificar o processo dos dados. Os métodos de detecção de "*outliers*" originais têm evoluído devido ao grande interesse dentro de áreas como a estatística e a ciência da computação.

A detecção dos "*outliers*" é uma tarefa crítica em muitos ambientes de segurança máxima, já que os "*outliers*" podem indicar condições anormais para as quais uma significativa degradação do rendimento pode ser detectada. Como exemplos disso um "*outlier*" pode

denotar um objeto estranho dentro de uma imagem como uma mina terrestre, pode mostrar um defeito na rotação de um motor em uma aeronave ou um problema de fluxo na tubulação interna de uma casa.

O caso contrário dos "*outliers*" são os "*inliers*". Os "*inliers*" são dados ou observações que se encontram em uma distribuição estatística e tem informação verdadeiramente representativa e significante de um sistema (Scheuren e Winkler, 1996). É importante conhecer o sistema de trabalho para saber quais pontos podem ser considerados como "*inliers*".

C - Método dos Mínimos Quadrados

O método dos mínimos quadrados é uma técnica de análises numérica que faz parte de um ramo chamado "otimização matemática", na qual tendo um conjunto de pares ordenados e uma família de funções, pretende-se encontrar a função que melhor se aproxime aos dados, utilizando o critério do mínimo erro quadrático.

O melhor ajuste para o conjunto de dados se consegue minimizando a soma dos quadrados das diferenças entre o valor estimado e os dados observados. Este método é aplicável quando se quer estimar os valores de uma variável (Y) considerando os valores de outra variável (X):

$$Y = \alpha + \beta X + E$$ Equação C-1

Onde α e β são constantes e E representa a variação de Y (o erro).

Utilizando o conceito de máxima verossimilhança se começa escolhendo um modelo matemático que aproxime o conjunto de dados (pares ordenados) para posteriormente medir a similaridade entre o conjunto dos dados do modelo e o conjunto dos dados obtidos baseando-se em uma função de custo (Arun, et al., 1987).

D - Decomposição em Valores Singulares (*SVD*)

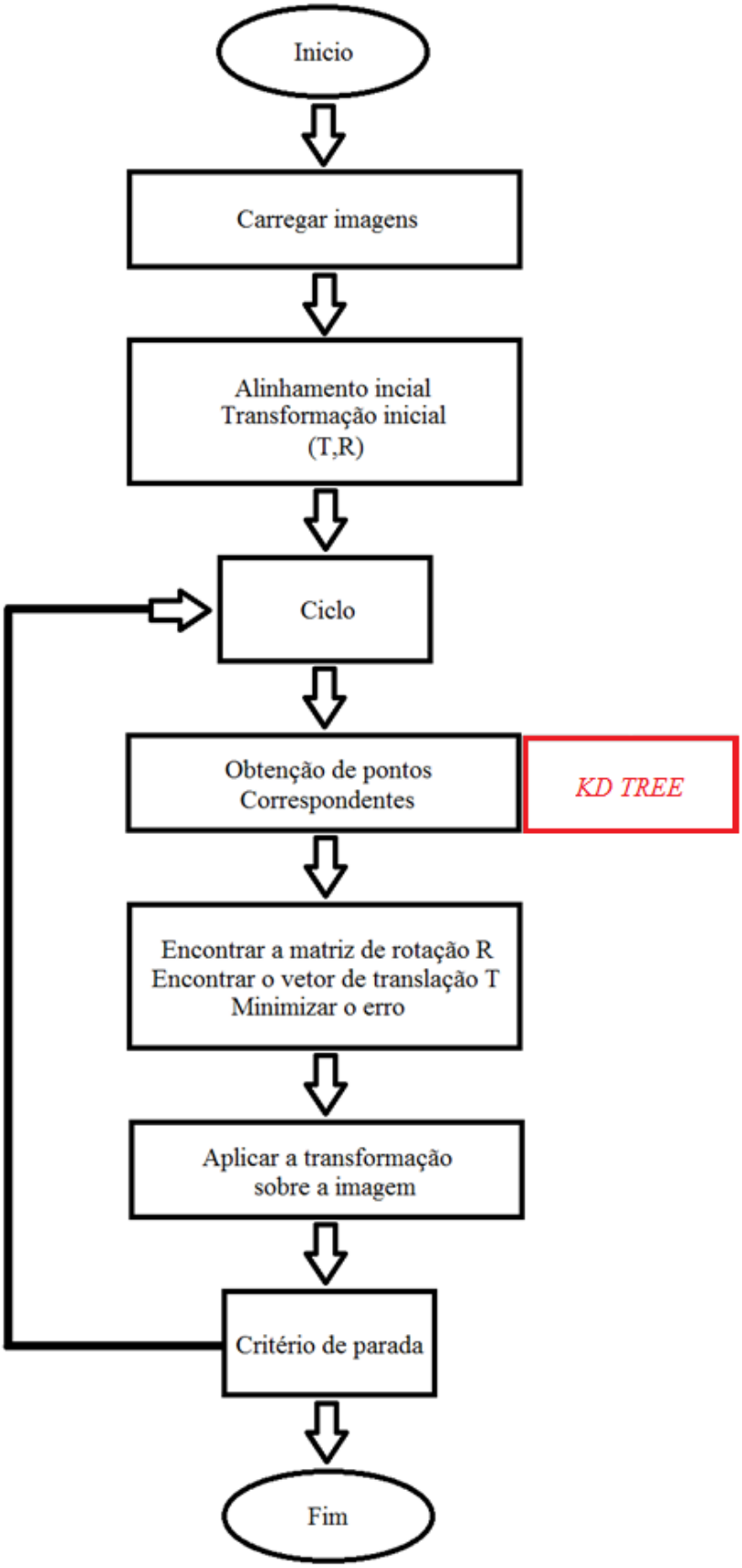
Uma decomposição em valores singulares ou *SVD* (das siglas em inglês *Singular Value*

Decomposition) de uma matriz real ou complexa é uma fatoração dela mesma, por exemplo, podemos calcular o *SVD* de uma matriz Z de tamanho m x n, real ou complexa se a fatoramos da seguinte forma:

$$Z=A\sum B^{*} \qquad \text{Equação D-1}$$

Onde m são as colunas, n são as fileiras, $\sum$ é uma matriz retangular diagonal de dimensão *m x n* (com números reais não negativos na diagonal), A é uma matriz unitária m x m (real ou complexa) e B* é uma matriz conjugada transposta unitária n x n que pode ser real ou complexa (Shlens, 2009).

DIAGRAMA DE BLOCOS *ICP-KD Tree*



CODIGO

```
[Imagem1 Imagem2 colunas fileras] = imagensmostras(nuvem1, nuvem2);
Trax = 0; Tray = 0; Traz = 0;
Tra = [Trax; Tray; Traz];
coordenadarotx = 0; coordenadaroty = 0; coordenadarotz = 0;
Rotx = [1 0 0; 0 cos(coordenadarotx) -sin(coordenadarotx); 0 sin(coordenadarotx) cos(coordenadarotx)];
Roty = [cos(coordenadaroty) 0 sin(coordenadaroty); 0 1 0; -sin(coordenadaroty) 0 cos(coordenadaroty)];
Rotz = [cos(coordenadarotz) -sin(coordenadarotz) 0; sin(coordenadarotz) cos(coordenadarotz) 0; 0 0 1];
Rot = Rotx*Roty*Rotz;
[RotnovaTranova Erro Time] = algoite(Imagem1, Imagem2, 1);
Imagemnova=repmat(Tranova, 1, fileras)+Rotnova * Imagem2;
figure;
plot3(Imagem1(1,:),Imagem1(2,:),Imagem1(3,:),'y.');
axisequal;
az = 0;
el = 90;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
title('Gráfico 1');
figure;
plot3(Imagem2(1,:),Imagem2(2,:),Imagem2(3,:),'b.');
axisequal;
az = 0;
el = 90;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
title('Gráfico 2');
figure;
plot3(Imagem1(1,:),Imagem1(2,:),Imagem1(3,:),'y.',Imagemnova(1,:),Imagemnova(2,:),Imagemnova(3,:),'b.');
axisequal;
az = 0;
el = 90;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
title('Gráfico 3');
figure;
plot3(Imagem1(1,:),Imagem1(2,:),Imagem1(3,:),'y.',Imagemnova(1,:),Imagemnova(2,:),Imagemnova(3,:),'b.');
axisequal;
az = 0;
el = 90;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
title('Gráfico 4');
figure;
plot3(Imagem1(1,:),Imagem1(2,:),Imagem1(3,:),'y.');
axisequal;
az = 270;
el = 0;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
title('Gráfico 1');
figure;
plot3(Imagem2(1,:),Imagem2(2,:),Imagem2(3,:),'b.');
axisequal;
az = 270;
el = 0;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
title('Gráfico 2');
figure;
plot3(Imagem1(1,:),Imagem1(2,:),Imagem1(3,:),'y.',Imagemnova(1,:),Imagemnova(2,:),Imagemnova(3,:),'b.');
axisequal;
az = 270;
el = 0;
view(az, el);
gridon;
xlabel('Eixo x'); ylabel('Eixo y'); zlabel('Eixo z');
```

```
title('Gráfico 3');
figure;
plot(Time,Erro,'--rs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','y','MarkerSize',10)
ylabel('Erro quadrático médio');
legend('"Força Bruta"');
function [Imagem1, Imagem2, colunas, fileras] = imagensmostras(nuvem1,nuvem2)
[X, Y, Z] = textread('nuvem.txt', '%f%f%f');
Imagem2 = [X(:), Y(:), Z(:)];
Imagem2 = Imagem2';
[X1, Y2, Z3] = textread('nuvem2.txt', '%f%f%f');
Imagem1 = [X1(:), Y2(:), Z3(:)];
Imagem1 = Imagem1';
[colunas,fileras]=size(Imagem2);
end
forite=1:numpontoslimite
correspon = zeros(1,colunas);
valordistanciaminimo = calculodist(imagem1,imagem2);
colunas = size(imagem1,2);
fileras = size(imagem2,2);
for contador=1:colunas
medistan=zeros(1,fileras);
for contador2=1:3
medistan = (medistan + (imagem2(:,contador2) - imagem1(contador2,contador)))^2;
end
[valordistanciaminimo(contador),correspon(contador)]=min(medistan);
end
valordistanciaminimo = sqrt(valordistanciaminimo);
centroidima2 = correspon(centroidima1);
valordistanciaminimo = valordistanciaminimo(centroidima1);
error(ite) = sqrt(sum(valordistanciaminimo.^2)/length(valordistanciaminimo));
colunas = size(imagem1,2);
fileras = size(imagem2,2);
desv1 = imagem2 - repmat(centroima1, 1, fileras);
desv2 = imagem1 - repmat(centroima2, 1, colunas);
ordena = desv2*transpose(desv1);
[U,S,V] = svd(ordena);
Rot = V*transpose(U);
if normal==1
[corresponminima]=corresponormal(imagem1,imagem2);
else
[corresponminima]=arvore(imagem1);
end
Tra = centroima1 - Rot*centroima2;
[Mudam1]=mudam(ite);
[Mudam2]=mudam2(ite,Tra);
MudamRot(ite+1,:,:) = mudam(ite,Rot);
MudamTra(ite+1,:,:) = mudam1(ite,Rot,Tra);
novoponto = MudamRot(:,:,ite+1) * imagem1 + repmat(MudamTra(:,:,ite+1), 1, novop);
error(ite+1) = erroquadraticomedio(imagem2(:,centroidima2), novoponto(:,centroidima1));
end
function [corresponvalordistanciaminimo] = corresponormal(imagem2, imagem1)
colunas = size(imagem1,2);
fileras = size(imagem2,2);
correspon = zeros(1,colunas);
valordistanciaminimo = zeros(1,colunas);
for contador=1:colunas
medistan=zeros(1,fileras);
for contador2=1:3
medistan=medistan+(imagem2(contador2,:)-imagem1(contador2,contador)).^2;
end
[valordistanciaminimo(contador),correspon(contador)]=min(medistan);
end
valordistanciaminimo = sqrt(valordistanciaminimo);
function [corresponvalordistanciaminimo] = arvore(imagem1)
tree = KDTreeSearcher(transpose(imagem1));
[corresponvalordistanciaminimo] = knnsearch(tree,transpose(imagem1));
correspon = transpose(correspon);
functionerror = erroquadraticomedio(ponto1,ponto2)
a = ponto1-ponto2;
b = power(a, 2),1;
dsq = sum(b);
error = sqrt(mean(dsq));
function mu=mudam(i,a)
a*MudamRot(i,:,:);
function mu2=mudam2(i,a,b)
a*MudamTra(i,:,:)+b;
```